LOS MICROPROCESADORES INTEL

8086/8088, 80186, 80286, 80386 y 80486
Arquitectura, programación e interfaces
Tercera Edición

BARRY B. BREY
DeVRY Institute of Technology

TRADUCCION:
Francisco Gutiérrez Noriega
C.P. y Perito Traductor

REVISION TECNICA:
M. en C. Agustín Suárez Fernández
Depto. Ingeniería Eléctrica
Universidad Autónoma Metropolitana-Iztapalapa



PRENTICE HALL HISPANOAMERICANA, S.A.
MEXICO • **ENGLEWOOD CLIFFS** • LONDRES • SYDNEY • TORONTO
NUEVA DELHI • TOKIO SINGAPUR • RIO DE JANEIRO

EDICION EN ESPAÑOL:
PRESIDENTE DIVISION LATINO AMERICA DE SIMON AND SCHUSTER: Raymundo Cruzado González
DIRECTOR GENERAL: Moisés Pérez Zavala
DIRECTOR DE EDICIONES: Alberto Sierra Ochoa
GERENTE DIVISION UNIVERSITARIA: Enrique Iván García Hernández
GERENTE EDITORIAL: José Tomás Pérez Bonilla
EDITOR: Luis Gerardo Cedeño Plascencia
SUPERVISOR DE TRADUCCION: Joaquín Ramos Santalla
SUPERVISION PRODUCCION: Julián Escamilla Liquidano

EDICION EN INGLES:
Editor: David Garza
Production Supervisor: Bookworks
Production Manager: Aliza Greenblatt
Text Designer: Debra A. Fargo
Cover Designer: Robert Freese
Illustrations: Academy Art Works, Inc.

BREY: *LOS MICROPROCESADORES INTEL 8086/8088, 80186, 80286, 80386 Y 80486 Arquitectura, programación e interfaces 3/Ed.*

Traducido del inglés de la obra: *The Intel Microprocessors 8086/8088, 80186, 80286, 80386 and 80486 Architecture, Programming, and Interfacing 3/Ed.*

ISBN 968-880-481-9

Miembro de la Cámara Nacional de la Industria Editorial, Reg. Núm 1524

ISBN 0-02-314250-2

Impreso en México/Printed in Mexico

FEB

PROGRAMAS EDUCATIVOS, S. A. DE C.V.
CALZ. CHABACANO No. 65, LOCAL A
COL. ASTURIAS,DELEG. CUAUHTEMOC,
C.P. 06850, MÉXICO, D.F.

EMPRESA CERTIFICADA POR EL
INSTITUTO MEXICANO DE NORMALIZACIÓN
Y CERTIFICACIÓN A.C., BAJO LA NORMA
ISO-9002: 1994/NMX-CC-004: 1995
CON EL No. DE REGISTRO RSC-048

2000 1998

Dedico este libro a mi amada esposa, Sheila, cuya paciencia y comprensión hicieron posible esta empresa.

PREFACIO

Este texto se escribió para el estudiante que está en un curso que requiere un conocimiento a fondo de la programación e interconexiones de la familia de microprocesadores Intel. Es un texto de consulta muy práctico para cualquier persona interesada en todos los aspectos de programación e interfaces de esta importante familia de microprocesadores. En la actualidad cualquier persona que actúa o se esfuerza por actuar en un campo de estudio en que se utilizan computadoras, debe entender la programación e interfaces del lenguaje ensamblador. Los microprocesadores Intel han logrado amplia aceptación en muchas áreas de la electrónica, comunicaciones, sistemas de control y, en particular, en los sistemas de computadoras de mesa.

ORGANIZACION Y COBERTURA

A fin de lograr un enfoque completo del aprendizaje, cada capítulo del texto empieza con un grupo de objetivos que son una breve definición del contenido del capítulo. Van seguidos por el cuerpo del texto, que incluye muchas aplicaciones de programación que ilustran los temas principales del capítulo. Al final del capítulo aparece un resumen con párrafos numerados, que tiene la doble función de guía para estudio y repaso de la información presentada en el capítulo. Al final, se presentan preguntas y problemas para promover la práctica y el ejercicio mental con los conceptos presentados en el capítulo.

Este libro contiene muchos ejemplos de programas con el empleo del programa Microsoft Macro Assembler, que ofrece la oportunidad de aprender la forma de programar la familia de microprocesadores Intel. El trabajo con los programas incluye las funciones de encadenador, biblioteca, macros, función DOS y funciones BIOS.

Además se presenta una descripción minuciosa de cada miembro de la familia, sistemas de memoria y diversos sistemas de E/S que incluyen: memoria en discos, ADC y DAC, USART, PIA, medidores de tiempo, controladores de teclado y exhibición, coprocesadores aritméticos y sistemas de exhibición en video. Con la ayuda de estos sistemas se aprende un método práctico para las interfaces de los microprocesadores.

ENFOQUE

Debido a que la familia de microprocesadores Intel es muy variada, este texto, al principio, se concentra en la programación en modo real, que es compatible con todas las versiones de la familia de microprocesadores Intel. Las instrucciones para cada miembro de la familia, que incluyen el 80386 y el 80486, se comparan y contrastan con las de los microprocesadores 8086/8088. Toda esta serie de microprocesadores son muy semejantes, lo cual permite aprender lo relativo a las versiones mas avanzadas, una vez que se comprende el 8086/8088 básico.

Además de la explicación completa de la programación y funcionamiento del microprocesador, en este texto también se explican la programación y el funcionamiento del coprocesador numérico (8087/80287/80387/80486/7). El coprocesador numérico funciona en un sistema para permitir acceso a los cálculos con punto flotante que son importantes en aplicaciones tales como sistemas de control, gráficas de video y diseño asistido por computadora (CAD). El coprocesador numérico permite a un programa accesar a complejas operaciones aritméticas que, en otra forma, sería difícil efectuar con la programación normal del microprocesador.

También se describen las conexiones en las patillas externas y las funciones de los microprocesadores 8086-80486. Las interfaces o interconexiones se explican primero con el empleo de los 8088/8086, con algunos de los componentes periféricos más comunes. Después de aprender lo básico, se concede gran importancia a lo relativo a los microprocesadores 80186/80188, 80386 y 80486. La descripción del 80286 debido a su semejanza con el 8086 y el 80386 es mínima, a fin de poder abarcar los 80386 y 80486 con todo detalle.

Con este enfoque, el funcionamiento del microprocesador y la programación con los miembros más modernos y avanzados de la familia, junto con la interconexión con todos los miembros de la familia, ofrece antecedentes prácticos y de trabajo de la familia de microprocesadores Intel. Al concluir un curso de estudios con este texto, usted estará en condiciones de:

1. Crear el software de control para una interconexión para aplicación con los microprocesadores 8086/8088, 80186/80188, 80286, 80386 u 80486. En general, el software que produzca también funcionará con todas las versiones de los microprocesadores. Este software también incluye aplicaciones basadas en DOS.
2. Programar con el empleo de la función DOS, el control del teclado, el sistema de exhibición de video y la memoria de disco en el lenguaje ensamblador.
3. Utilizar las funciones BIOS para controlar el teclado, exhibición y otros componentes en el sistema de la computadora.
4. Producir un software en que se utilicen ganchos para interrupción y teclas con corriente para obtener acceso para terminar y anclar el software residente.
5. Programar el coprocesador numérico (80287/80387) para resolver ecuaciones complejas.
6. Explicar las diferencias entre los miembros de la familia y destacar las características de cada miembro.
7. Describir y usar el funcionamiento en modo real y protegido de los microprocesadores 80286, 80386 y 80486,
8. Interconectar la memoria y los sistemas E/S con el microprocesador.
9. Suministrar una comparación detallada y completa de todos los miembros de la familia, su software y su interconexión con el hardware.
10. Explicar el funcionamiento de los sistemas de disco y de video.

RESUMEN DEL CONTENIDO

En el capitulo 1 se presenta a la familia de microprocesadores Intel y se subraya el sistema de computadora basado en microprocesador. Este primer capítulo sirve como introducción del microprocesador, su historia, su funcionamiento y los métodos utilizados para almacenar datos en un sistema basado en microprocesador. También se estudian el modelo de programación y la arquitectura del sistema. Se explica el funcionamiento tanto en modo real como protegido en este capítulo que sirve de introducción. Una vez que se capta la comprensión de la máquina básica, en los capítulos 2 a 5 se explica como actúa cada instrucción en la familia de microprocesadores Intel. Conforme se explican las instrucciones, se presentan aplicaciones sencillas para ilustrar el funcionamiento de las instrucciones y para establecer conceptos básicos para la programación.

Una vez establecida la base para la programación, en el capítulo 6 se describen aplicaciones con el empleo del programa ensamblador. Estas aplicaciones incluyen la programación con el empleo de llamadas a las funciones de DOS y BIOS. Se explican los archivos de disco así como el funcionamiento del teclado y el video en un sistema de computadora personal. En este capítulo se suministran las herramientas requeridas para crear virtualmente cualquier programa en un sistema de computadora personal. También se introduce el concepto de ganchos para interrupción y llaves con corriente

En el capitulo 7 se presenta la familia de 8086/8088 como la base para aprender la interconexión básica de la memoria y de E/S que aparecen en capítulos posteriores. En este capítulo se presenta el sistema con búfer (memoria intermedia) así como la sincronización del sistema.

En el capítulo 8 se presentan detalles completos de la interconexión de la memoria con el empleo de codificadores integrados y dispositivos de lógica programables. Se ilustran la paridad así como los sistemas de memoria dinámica. Se ofrecen los sistemas de memoria de 8, 126 y 32 bits para poder interconectar los microprocesadores 8086-80486 con la memoria.

En el capítulo 9 se estudia en forma detallada la interconexión o interfaz E/E que incluye PIA, medidores de tiempo, interconexiones entre teclado y exhibición, los USART y los convertidores ADC y DAC. También se describe la interconexión de los motores de CC y los escalonadores o de velocidad gradual.

Una vez que se han entendido estos componentes básicos para E/S y sus interconexiones con el microprocesador, en los capitulos 10 y 11 se dan detalles de las técnicas avanzadas para E/S que incluyen las interrupciones y el acceso directo a la memoria (DMA). Las aplicaciones incluyen una interconexión para la impresora, reloj en tiempo real, memoria de disco y sistemas de video.

En el capítulo 12 se detallan el funcionamiento y la programación de la familia de coprocesadores aritméticos 8087-80387. En la actualidad, pocas aplicaciones funcionan con eficiencia sin la potencia del coprocesador aritmético.

En los capítulos 13 y 14 se ofrecen detalles de los avanzados microprocesadores 80186/80188-80486. En estos capitulos se exploran las diferencias entre los 8086/88 y sus activaciones o incrementos. La memoria caché así como la memoria de intercalación y de ráfaga también se describen con los microprocesadores 80386 y 80486. También se describen el manejo o administración de la memoria y la paginación de la memoria.

Se incluyen apéndices para acrecentar la aplicación del texto e incluyen:

A. Una lista completa de las llamadas de función a DOS INT 21H. En este apéndice también se detalla el uso del programa ensamblador y de muchas de las llamadas de función NBIOS incluso la función de llamada INT 10H a BIOS.

B. Lista completa de todas las instrucciones para 8086/8088/80286/80386/80486 que incluyen muchos ejemplos de ilustraciones y para codificación de la máquina en hexadecimal así como información para sincronización del reloj.
C. Una lista compacta de todas las instrucciones que cambian los bits indicadores.
D. Detalle de las conexiones en patillas externas de las normas de bus de computadora personal para poder interconectarla con sistemas adicionales de hardware.
E. Respuestas a las preguntas y problemas con número par.

B.B.B.

CONTENIDO

CAPITULO 1

Introducción al microprocesador

INTRODUCCION

En este capítulo se presenta la familia de microprocesadores Intel. Este importante grupo de microprocesadores incluye los 8086, 8088, 80186, 80286, 80386 y 80486, que son una evolución de los primeros microprocesadores de 4 y 8 bits. En el capítulo, se describen la arquitectura y la estructura de memoria del microprocesador y se ofrece el conjunto de instrucciones de los microprocesadores Intel.

También se presentan los formatos utilizados en el microprocesador para almacenamiento de datos. Estos formatos incluyen enteros binarios almacenados como números con signo y sin signo, como bytes, palabras y dobles palabras, datos en BCD, datos con punto decimal flotante y datos en ASCII.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo el lector podrá:

1. Describir la evolución del microprocesador.
2. Explicar el funcionamiento interno del microprocesador detallando la arquitectura paralela (pipeline), la "cola" de códigos de instrucción y diversas estructuras internas.
3. Detallar la estructura de la memoria y el arreglo de registros internos del microprocesador.
4. Definir los términos de segmento y desplazamiento para una dirección de memoria.
5. Calcular la dirección efectiva para el siguiente paso del programa, con el empleo del contenido de los registros del apuntador de instrucciones (IP) y del segmento de código (CS).
6. Mostrar cómo se almacenan los datos en la memoria de los siguientes tipos de datos de enteros y sin signo: byte, palabra y doble palabra.
7. Definir el modo de funcionamiento real y protegido conforme se aplica a los miembros más avanzados de la familia.
8. Presentar un panorama del grupo de instrucciones de la familia Intel.

1–1 LA EVOLUCION DEL MICROPROCESADOR

Antes de comentar los microprocesadores modernos, se debe primero entender qué fue lo que puso a estos dispositivos en primer plano. La historia dice que los antiguos babilonios empezaron a usar el *ábaco*, (calculadora primitiva hecha con cuentas o esferas ahuecadas), alrededor del año 500 a. de JC. Con el tiempo, esta sencilla calculadora, estimuló a la humanidad para perfeccionar una maquinaria calculadora en que se utilizaban engranes y ruedas (Blas Pascal en 1642). Se continuaron los progresos con las gigantescas máquinas computadoras de las décadas de 1940 y 1950, construidas con relevadores y tubos de vacio (bulbos). Más adelante, se utilizaron los transistores y los componentes electrónicos de estado sólido para construir las poderosas computadoras de la década de 1960. Con el advenimiento de los circuitos integrados se llegó al perfeccionamiento del microprocesador y de los sistemas de microcomputadoras.

El microprocesador de 4 bits

En 1971, Intel Corporation y el talento creativo de Marcian E. Hoff lanzaron el primer microprocesador: el 4004, de 4 bits. Este controlador integrado, programable en un solo encapsulado era insuficiente, según las normas actuales, porque sólo direccionaba 4096 localidades de 4 bits en la memoria. El 4004 contenía un conjunto de instrucciones que ofrecían sólo 45 instrucciones diferentes. Como consecuencia, el 4004 sólo se podía emplear en aplicaciones limitadas, como en los primeros juegos de video y en controladores pequeños basados en microprocesadores. Cuando surgieron aplicaciones más complejas para el microprocesador, el 4004 resultó inadecuado.

El microprocesador de 8 bits

Más tarde, en 1971, al percatarse que el microprocesador era un producto viable para comercialización, Intel Corporation produjo el 8008, el primer microprocesador de 8 bits. El tamaño ampliado de la memoria (16K 8) y las instrucciones adicionales (un total de 48) en este nuevo microprocesador brindaron la oportunidad de muchas aplicaciones más avanzadas (1K es igual a 1024 y un *byte* es un número de 8 bits).

Conforme los ingenieros desarrollaban usos más demandantes para el microprocesador, la memoria y el juego de instrucciones más o menos pequeños del 8008 pronto limitaron su utilidad. Por tanto, en 1973, Intel Corporation introdujo el 8080, el primero de los microprocesadores modernos de 8 bits. Pronto, otras empresas empezaron a lanzar sus propias versiones de los procesadores de 4 y de 8 bits. En la tabla 1-1 se enumeran muchos de estos primeros microprocesadores.

TABLA 1–1 Primeros procesadores de 8 bits

Fabricante	*Número de pieza*
Fairchild	F-8
Intel	8080
MOS Technology	6502
Motorola	MC6800
National Semiconductor	IMP-8
Rockwell International	PPS-8

¿Pero qué tenía de especial el 8080? No sólo direccionaba más en la memoria y ejecutaba más instrucciones, y con diez veces más rapidez que el 8008. Una suma que tardaba 20 µs en un sistema basado en un 8008, sólo necesitaba 2.0 µs en un sistema basado en un 8080. Además, el 8080 era compatible con la lógica transistor-transistor (TTL) lo cual significaba que se podía interconectar con componentes de lógica TTL estándar. Todas estas ventajas introdujeron la era del 8080 y la época siempre en expansión del *microprocesador*.

Intel Corporation introdujo en 1977, una nueva versión del 8080: el 8085. Aunque sólo ligeramente más avanzado que el 8080, el 8085 direcciona la misma cantidad de memoria, ejecuta más o menos el mismo número de instrucciones y suma en 1.3 µs en lugar de 2.0 µs. Las principales ventajas del 8085 son el generador de reloj y el controlador del sistema integrados, que eran componentes externos en un sistema basado en el 8080. Tan sólo Intel ha vendido más de 100 millones de piezas del microprocesador 8085. Otras empresas, como NEC, AMD, Toshiba y Hitachi también fabrican, bajo licencia, una versión del microprocesador 8085. Estas características han hecho del 8085 uno de los microprocesadores Intel más conocidos.



El microprocesador de 16 bits

En 1978, Intel Corporation lanzó el microprocesador 8086 y más o menos un año más tarde, el 8088. Ambos dispositivos son microprocesadores de 16 bits, que ejecutan instrucciones en escasos 400 ns; una gran mejoría en relación con la velocidad de ejecución del 8085. El 8086 y el 8088 tienen también capacidad para direccionar a un 1M byte (8 bits) o una memoria de 512K palabra (16 bits de ancho). Las velocidades más altas de ejecución y el tamaño de memoria más grande, permitieron al 8086 y al 8088 sustituir a minicomputadoras pequeñas en muchas aplicaciones.

Una necesidad importante que aceleró la evolución del microprocesador de 16 bits fue la multiplicación y la división por hardware. Estas funciones no están disponibles en la mayor parte de los microprocesadores de 8 bits, excepto el motorola MC6809, que puede multiplicar, pero no dividir. Pero el microprocesador de 16 bits evolucionó también por otras razones. Ofrece un espacio direccionable de memoria más grande que el microprocesador de 8 bits, lo cual le permite efectuar operaciones muy complejas para las cuales no hay espacio en 64K bytes de memoria. El 8086 y el 8088 tienen un gran número de registros internos, accesibles a 200 ns, por comparación con los 80 que se necesitan para llegar hasta un registro en un microprocesador de 8 bits. Estos registros adicionales permiten escribir software con mucha más eficiencia. Asimismo, los programas de aplicación del software (sistemas administrativos basados en datos, listados, procesadores de palabras y verificadores de ortografía) empezaron a requerir más de los 64K bytes de memoria disponibles en el microprocesador de 8 bits. El momento había llegado para el microprocesador de 16 bits.

La evolución del microprocesador de 16 bits no terminó en el 8086 y el 8088, sino que continuó con la introducción del 80186, una versión altamente integrada del 8086. El microprocesador 80186 de 16 bits es uno de los más conocidos de Intel. El 80186 se utiliza en muchas aplicaciones de sistemas de control, pero no como el microprocesador principal en los sistemas de computadoras personales. Si se encuentra el 80186 en una computadora personal es sólo en una tarjeta enchufable en la tarjeta madre que podría controlar una memoria en disco duro o una interface para comunicaciones.

El más reciente microprocesador de 16 bits producido por Intel es el 80286, una versión mejorada del 8086, que contiene una unidad de administración de memoria y direcciona a una memoria de 16M en lugar de 1M byte. La velocidad de reloj del 80286 se ha aumentado también

TABLA 1–2 Microprocesadores Intel

Número de pieza	*Ancho del canal de datos*	*Tamaño de memoria*
8048	8	2K interna
8051	8	8K interna
8085A	8	64K
8086	16	1M
8088	8	1M
8096	16	8K interna
80186	16	1M
80188	8	1M
80286	16	16M
80386DX	32	4G
80386SL	16	32M
80386SX	16	16M
80486DX	32	4G
80486SX	32	4G
Pentium*	32/64	4G

*Pentium es una marca registrada de Intel Corporation.

a 16 MHz, en las últimas versiones producidas por Intel. La versión básica del 8086 y del 8088 ejecutaba hasta 2.5 MIP (*millones de instrucciones por segundo*), en tanto la versión básica del 80286 ejecuta hasta 8 MIP.

El microprocesador de 32 bits

La versión más reciente del microprocesador es el de 32 bits (consulte en la tabla 1-2 la lista de todos los microprocesadores Intel). Intel produce en la actualidad dos versiones principales: el 80386 y el 80486. El 80386 fue el primer microprocesador de 32 bits producido por Intel, cuya principal ventaja es una frecuencia de reloj mucho más alta (33 MHz en el 80386 y 66 MHz en la versión de doble reloj del 80486), así como un espacio mucho mayor en la memoria (4G bytes).

El microprocesador 80486 contiene básicamente un 80386 mejorado, un coprocesador aritmético (para la versión DX del 80486) y una memoria caché interna de 8K bytes. El 80386 ejecuta muchas instrucciones en 2 ciclos de reloj, mientras que el 80486 ejecuta muchas instrucciones en un solo ciclo de reloj. Estas mejoras, combinadas con un reloj de 66 MHz (80486DX2) permiten que las instrucciones se ejecuten a 54 MIP, de acuerdo con Intel Corporation. Esto puede compararse con el 8085, presentado 12 años antes que el 80486, que ejecutaba las instrucciones a una velocidad de alrededor de 0.5 MIP. Estas mejoras en la velocidad continuaron con las versiones más nuevas del microprocesador de 32 bits conforme sea disponible. La próxima generación (Pentium) promete lograr velocidades de 100 MIP.

1–2 ARQUITECTURA BASICA DEL MICROPROCESADOR

La programación e interface eficiente requiere una comprensión clara de la arquitectura básica de la familia de microprocesadores y de los sistemas de computación basados en microprocesadores

de Intel. En esta sección se presenta una descripción detallada de la arquitectura básica de estos microprocesadores.

Arquitectura interna básica

Los microprocesadores modernos, igual que los anteriores, buscan instrucciones en la memoria, pero lo hacen en una forma totalmente nueva. Los microprocesadores modernos están estructurados de modo que contengan muchas más unidades internas de procesamiento, cada una de las cuales efectúa una tarea específica. (Téngase en cuenta que cada una de estas unidades de procesamiento es, en realidad, un microprocesador para fines especiales.) Esto significa que el microprocesador moderno puede a menudo procesar cierto número de instrucciones en forma simultánea, en diversas etapas de la ejecución. Esta capacidad recibe a menudo el nombre de *paralelismo*.

En la figura 1-1(a) se ilustra el funcionamiento normal de un 8085, que es típico de la mayor parte de los microprocesadores de 8 bits. Nótese que las instrucciones se recuperan de la memoria con una operación de lectura. Luego, mientras el 8085 ejecuta la instrucción, el sistema de memoria está ocioso. La familia de microprocesadores Intel, empezando con el 8086 y el 8088, aprovecha este tiempo de memoria ociosa buscando con anticipación la siguiente instrucción mientras ejecuta la actual. Esto acelera la ejecución total de un programa.

En la figura 1-1(b) se ilustra la secuencia de eventos en el microprocesador 80486. Se notará que el canal está casi siempre ocupado. Se verá también que el 80486 tiene más de una unidad

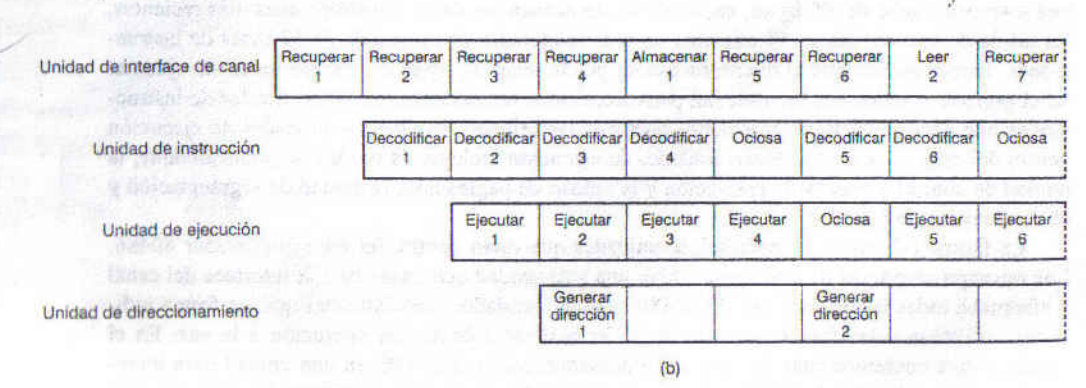
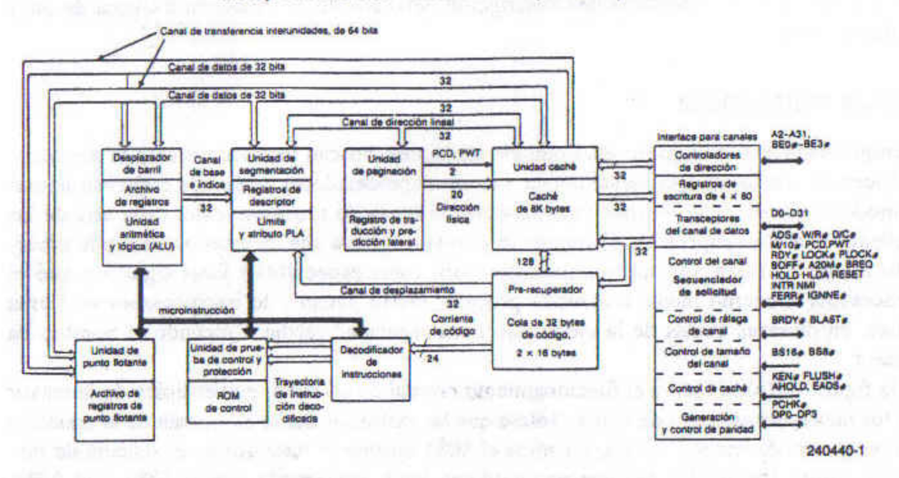


FIGURA 1-1 (a) Funcionamiento de un microprocesador antiguo como el 8085A; (b) Funcionamiento del microprocesador 80486 con arquitectura paralela (pipeline).



iRMX, iRMK, 386, 387, 486, i486 son marcas registradas de Intel Corporation.
*MS-DOS® es una marca registrada de Microsoft Corporation.
**OS/2MR es una marca registrada de Microsoft Corporation.
***UNIX MR es una marca registrada de AT&T.

FIGURA 1–2 La estructura interna del microprocesador 80486. (Cortesía de Intel Corporation.)

interna. Cada unidad está destinada a funcionar en *paralelo* con instrucciones en diversas fases de la ejecución.

La unidad de interface del canal (BIU), es la que tiene la función de leer las instrucciones y de leer o escribir datos entre el microprocesador y la memoria. La salida de la BIU se conecta con una memoria caché de 8K bytes, en donde se almacenan los datos e instrucciones más recientes. La salida de la memoria caché alimenta un prerrecuperador con una cola de 32 bytes de instrucciones. Esto significa que el microprocesador puede tener 32 bytes de instrucciones sin ejecutar en el prerrecuperador. En la salida del prerrecuperador se encuentra un decodificador de instrucciones que decodifica las instrucciones para que las utilicen las diversas unidades de ejecución dentro del microprocesador. Estas unidades de ejecución incluyen la unidad de punto flotante, la unidad de control y prueba de protección y la unidad de paginación, la unidad de segmentación y el desplazador de barril, y la ALU.

La figura 1-2 ilustra la multitud de unidades que están dentro del microprocesador 80486. Los microprocesadores más antiguos tenían una sola unidad que controlaba la interface del canal y efectuaba todas las operaciones. El 80486 contiene unidades especializadas que, en forma individual, efectúan estas funciones a fin de poder realizar más de una operación a la vez. En el futuro, estará contenido más de un microprocesador, como el 80486, en una unidad para incrementar, adicionalmente, la velocidad del sistema para las secuencias de instrucciones que no son dependientes.

Arquitectura del sistema

En la figura 1-3 se ilustra la arquitectura del sistema de un microprocesador moderno. Se verá que las comunicaciones entre el sistema y el microprocesador ocurren por medio de estos canales: direcciones, datos y control. El *canal de direcciones* suministra una dirección en la memoria para la memoria del sistema o el espacio de *entrada/salida* (E/S) para los dispositivos de E/S del sistema. El *canal de datos* transfiere éstos entre el microprocesador y la memoria y los dispositivos de E/S conectados en el sistema. El *canal de control* suministra señales de control que hacen que en la memoria o el espacio de E/S efectúen una operación de lectura o de escritura. Las señales de control que hacen posible la lectura o escritura en la memoria o el espacio de E/S varían muy poco entre un miembro y otro de la familia.

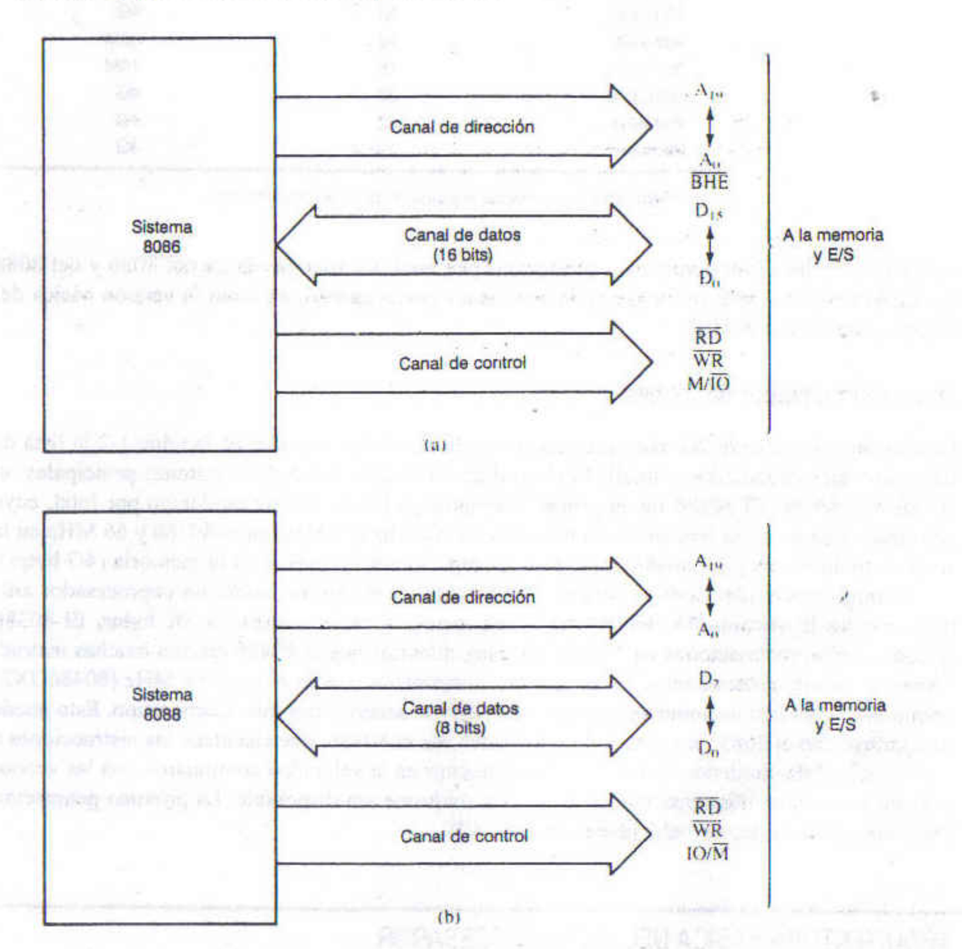


FIGURA 1–3 Sistemas básicos 8086 y 8088. (a) El sistema 8086 ilustra el canal de datos de 16 bits, el canal de direcciones de 20 bits y el canal de control. (b) El sistema del 8088 ilustra el canal de datos de 8 bits, el canal de direcciones de 20 bits y el canal de control.

TABLA 1–3 Comparación de dirección, datos y velocidades de reloj de los miembros de la familia Intel

Microprocesador	*Ancho del canal de datos (bits)*	*Ancho del canal de direcciones (bits)*	*Reloj (MHz)*
8086	16	20	5
8088	8	20	5
80186	16	20	6
80286	16	24	8
80386DX	32	32	16
80386SX	16	24	16
80386SL	16	25	16
80486DX	32	32	25
80486SX	32	32	20

Un examen cuidadoso de la figura 1-3 revelará que el ancho del canal de datos y el de direcciones varían entre una versión y otra del microprocesador. En la tabla 1-3 se enumeran estas variaciones para comparación y también la velocidad de reloj cuando hizo su aparición el microprocesador. Después de la versión inicial de cada microprocesador, se pusieron a disposición versiones con frecuencias de reloj más altas.

El ancho del canal de datos determina cuántos bytes se transfieren a la vez entre el microprocesador y la memoria, mientras que el ancho del canal de direcciones determina cuánta memoria direcciona el microprocesador. En general, cuanto más ancho sea el canal de datos, más rápido es el microprocesador. En las futuras generaciones de microprocesadores y de microcomputadoras se tendrán canales de datos de 64 y, quizá, de 128 bits.

1–3 LA MEMORIA Y EL MICROPROCESADOR

El espacio de direccionamiento de un sistema basado en un microprocesador, se denomina memoria lógica o memoria física. La estructura de la memoria lógica es diferente, en casi todos los casos, que la estructura de la memoria física. La memoria lógica es el sistema de memoria tal como lo ve el programador, mientras que la memoria física es la estructura real en el hardware en el sistema de memoria.

Memoria lógica

El espacio básico de la memoria lógica es el mismo en todos los microprocesadores Intel. La memoria lógica se numera por bytes. En la figura 1-4 se ilustra el mapa de la memoria lógica de todos los miembros de la familia Intel. Se verá que la única diferencia es que algunos miembros contienen más memoria que otros. Además, se debe tener en cuenta que la memoria física puede diferir de la memoria lógica en muchos sistemas.

La memoria lógica del 8086, 8088 y 80186 empieza en la localidad 00000H y llega hasta la FFFFFH. Este intervalo de direcciones especifica el 1M de bytes de memoria disponible en estos sistemas. La memoria lógica del 80286 y el 80386SX empieza en la localidad de memoria 000000H y llega hasta la ubicación FFFFFFH para tener un total de 16M bytes de memoria. El

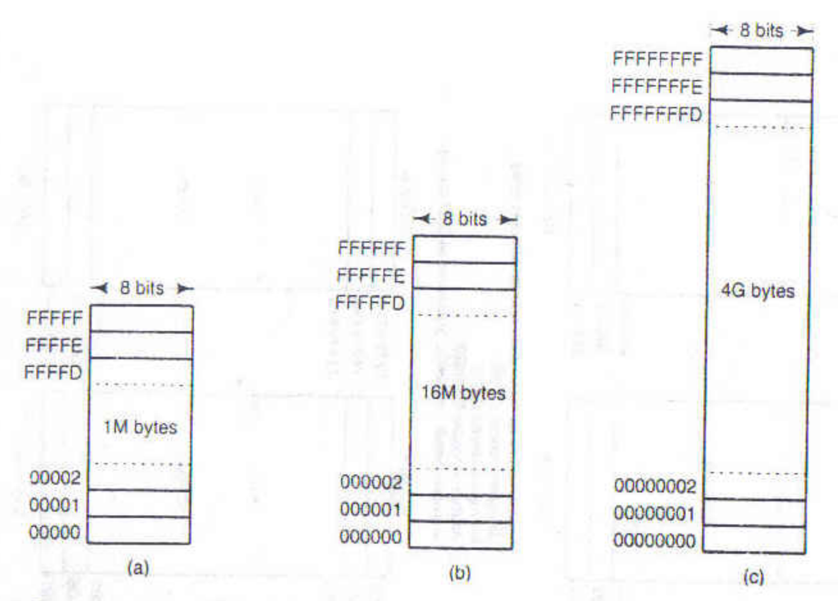


FIGURA 1–4 El mapa de memoria lógica de los microprocesadores: (a) 8086/8088/80186; (b) 80286/80386[redacted] (c) 80386DX y 80486.

80386SL contiene 32M bytes de memoria que empiezan en la localidad 0000000H y terminan en la 1FFFFFFH. El 80386DX, 80486SX y 80486DX contienen memoria que empieza en la localidad 00000000H y termina en la FFFFFFFFH, para un total de 4G bytes de memoria (*1G = 1,024M y 1M = 1024K*).

Cuando estos microprocesadores direccionan una palabra de 16 bits en la memoria, se accesan dos bytes consecutivos. Por ejemplo, la palabra en la localidad 00122H se almacena en los bytes 00122H y 00123H; el byte menos significativo se almacena en la localidad 00122H. Si se accesa a una palabra doble de 32 bits, esta palabra doble la contienen cuatro bytes consecutivos. Por ejemplo, la doble palabra almacenada en la localidad 00120H se almacena en los bytes 00120H, 00121H, 00122H y 00123H; el byte menos significativo se almacena en 00120H y el byte más significativo en 00123H.

Memoria física

Las memorias físicas de los miembros de la familia Intel difieren en ancho. La memoria del 8088 es de 8 bits de ancho; las memorias del 8086, 80186, 80286 y 80386SX tienen 16 bits de ancho; las memorias del 80386DX y 80486 son de 32 bits de ancho. Para la programación, no hay diferencia en el ancho de la memoria porque la memoria lógica siempre es de 8 bits de ancho; pero, como se puede ver en la figura 1-5, hay una diferencia para el diseñador del hardware.

La memoria está organizada en bancos de memoria en todas las versiones del microprocesador, excepto el 8088 que tiene un solo banco de memoria. Un *banco de memoria* es una sección de 8

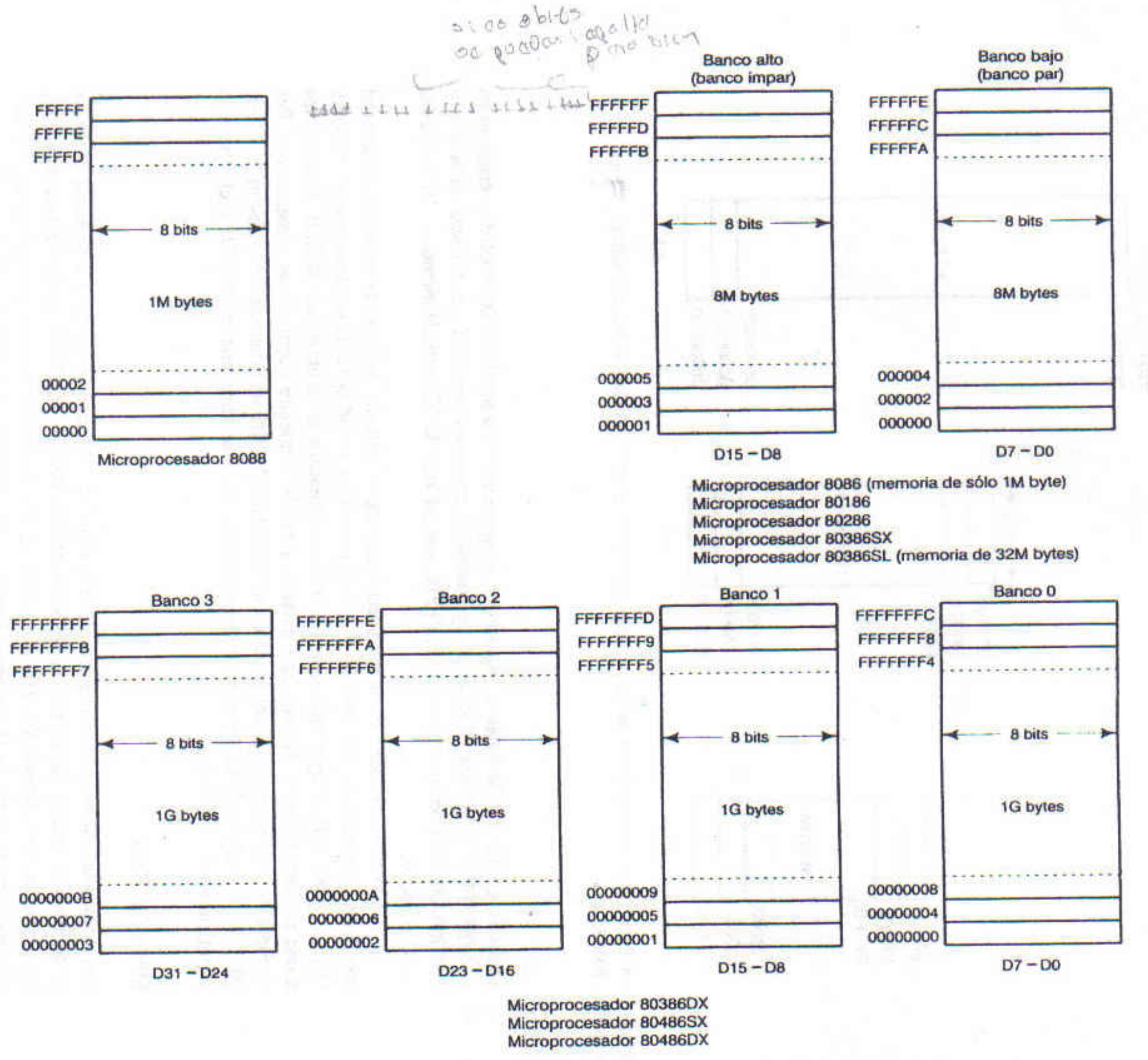


FIGURA 1–5 Los sistemas de memoria física de la familia de microprocesadores 8086-80486.

bits de ancho. Los microprocesadores de 16 bits tienen dos bancos de memoria para formar una sección de memoria de 16 bits de ancho, a la cual se direcciona por bytes o por palabras. Los microprocesadores de 32 bits tienen cuatro de bancos de memoria, pero se les direcciona como bytes, palabras o dobles palabras.

La memoria en la computadora personal

Cualquier estudio de la familia Intel requiere entender la estructura de la memoria de la computadora personal. Debido a que la computadora personal original estuvo basada en el microprocesador 8088, se considera que su memoria principal tiene una longitud de 1M byte. Esta memoria principal se llama *memoria real*. Nunca se toleró una ampliación de la memoria hasta que LIM (Lotus-Intel-Microsoft) crearon un estándar para un *sistema de memoria ampliada* (EMS).

La memoria ampliada se colocó en un *marco de página* vacío (64K bytes) ubicado entre la memoria de sólo lectura (ROM) del BIOS (Sistema básico de entrada/salida) en el sistema. Con este marco de página de 64K bytes, el estándar LIM permite acceso a un número ilimitado de paginas de memoria ampliada de 64K bytes, si bien el acceso a esas paginas es lento. El sistema de memoria ampliada se volvió obsoleto con el advenimiento del microprocesador 80286 y otros más nuevos, aunque todavía está soportado por los antiguos sistemas basados en 8086 y 8088.

Con los nuevos 80286, 80386 y 80486 se pueden direccionar memorias por encima del primer byte 1M. Esta memoria adicional, llamada *sistema de memoria extendida* (XMS) contiene 15M bytes adicionales en el sistema de 80286 y 80386SX, y 4,095M bytes en los sistemas de 80386DX y 80486. El sistema XMS ha sustituido al sistema EMS de las primeras computadoras personales.

En la figura 1-6 se muestra un mapa de memoria de una computadora personal con la memoria etiquetada por zonas. Los primeros 640K bytes del sistema de memoria en todas las computa-

FIGURA 1-6 El mapa de memoria de un sistema de computadora personal.

doras personales, se llama el *área de programa transitorio* (TPA). El TPA contiene una memoria RAM (*lectura y escritura*) para almacenar las aplicaciones de software, el sistema operativo y diversos programas que controlan los dispositivos de E/S. Después del TPA está almacenada la zona de sistemas que contiene varios BIOS (*sistema básico de entrada/salida*) para controlar el sistema, una RAM de video y zonas abiertas que se pueden utilizar para un marco de página de EMS y con opciones instalables en el sistema de la computadora. Encima de esta zona de memoria de 1M byte, está el sistema de memoria extendida, que tiene sistemas para discos en cacheo y otros segmentos de datos definidos por el sistema operativo.

1–4 EL MODELO DE PROGRAMACION

La programación requiere un claro entendimiento de la estructura de los registros internos de la familia de microprocesadores Intel. En esta sección se describe la estructura de los registros del microprocesador y se explica la forma en que se direcciona la memoria por medio de los registros de segmentos y de los desplazamientos de dirección.

En la figura 1-7 se ilustra el arreglo de registros internos del microprocesador. Este arreglo se aplica a todas las versiones del microprocesador. Tenga en cuenta que las partes sombreadas sólo están disponibles en los microprocesadores 80386 y 80486. El arreglo de registros internos consta de tres grupos de registros: registros de uso general, apuntadores y registros de índice y registros de segmentos. Además de esos grupos, hay también un registro de banderas que señala las condiciones respecto al funcionamiento de la unidad aritmética y lógica (ALU).

Registros de propósito general

Los *registros de propósito general* se utilizan en la forma en que desee el programador. Cada registro para uso general se puede direccionar como un registro de 32 bits (EAX, EBX, ECX y EDX), como uno de 16 bits (AX, BX, CX y DX) o como uno de 8 bits (AH, AL, BH, BL, CH, CL, DHL y DL). Se debe tener en cuenta que sólo el 80386 y el 80384 contienen el grupo de registros de 32 bits. En algunas de las instrucciones que se explican en capítulos posteriores también se utilizan los registros de uso general para tareas específicas. Por esta razón, a cada uno se le da su nombre (Acumulador, Base, Contar y Datos). En lenguaje ensamblador, al registro de uso general siempre se le denomina con combinaciones de dos o de tres letras. Por ejemplo, al acumulador se le denomina EAX, AX, AH o AL.

Las funciones primarias de los registros de propósito general incluyen:

AX (*Acumulador*): a menudo conserva el resultado temporal después de una operación aritmética o lógica. También se le direcciona como EAX, AH o AL.

BX (*Base*): a menudo conserva la dirección base (desplazamiento) de los datos que hay en la memoria o la dirección base de una tabla de datos referenciados por la instrucción para convertir (XLAT). También se le direcciona como, EBX, BH o BL.

CX (*Contador*): contiene el conteo de ciertas instrucciones para corrimientos (CL) y rotaciones del número de bytes (CX) para las operaciones repetidas de cadena y un contador (CX o ECX) para la instrucción LOOP. También se le direcciona como ECX, CH o CL.

FIGURA 1–7 Arreglo de registros internos (modelo de programación) de todas las versiones de los microprocesadores.



Nombres de 32 bits	Nombres de 16 bits (Nombres de 8 bits)	Nombre
EAX	AH AX AL	Acumulador
EBX	BH BX BL	Indice base
ECX	CH CX CL	Contador
EDX	DH DX DL	Datos
ESP	SP	Apuntador de pila
EBP	BP	Apuntador de base
EDI	DI	Indice destino
ESI	SI	Indice fuente
← 32-bits →	← 16-bits →	
EIP	IP	Apuntador de instrucciones
EFLAGS	FLAGS	Banderas
	CS	Código
	DS	Datos
	ES	Extra
	SS	Pila
	FS	
	GS	

Notas:
1. Las partes sombreadas no están disponibles en los microprocesadores 8086,8088 u 80286.
2. No se dan nombres especiales a los registros FS y GS.

DX (*Datos*): es un registro de uso general que también contiene la parte más significativa de un producto después de una multiplicación de 16 o de 32 bits; la parte más significativa del dividendo antes de la división y el número de puerto de E/S para una instrucción variable de E/S. También se le direcciona como EDX, DH o DL.

Registros apuntadores e índices

Aunque los registros apuntadores e índices son también de uso general, se utilizan más a menudo para apuntar a la localidad de la memoria que contiene los datos del operando de muchas instrucciones.

Los registros apuntadores e índices incluyen:

SP (*Apuntador de pila*). Se emplea para direccionar datos en una pila de memoria de LIFO (último en entrar, primero en salir). Esto ocurre más a menudo cuando se ejecutan las instrucciones PUSH y POP cuando se llama (CALL) o se regresa (RET) de una subrutina desde un programa principal. Este registro es también el registro ESP de 32 bits.

BP (*Apuntador de base*). Es un apuntador de uso general que se utiliza a menudo para direccionar a una matriz de datos en una pila de memoria. Este registro es también el registro EBP de 32 bits.

SI (*Indice de fuente*). Se emplea para direccionar datos fuente en forma indirecta para utilizarlos con las instrucciones de cadenas o arreglos. Este registro es también el registro ESI de 32 bits.

DI (*Indice de destino*). Se suele emplear para direccionar datos destino en forma indirecta, para utilizarlos con las instrucciones de cadenas o arreglos. Este registro es también el registro EDI de 32 bits.

IP (*Apuntador de instrucciones*). Se utiliza siempre para direccionar a la siguiente instrucción que va a ejecutar el microprocesador. Para formar la localidad real de la siguiente instrucción se suma el contenido de IP con CS (por) 10H, como se describe en la siguiente sección. Este registro es también el registro EIP de 32 bits.

A menudo se hace direccionamiento indirecto de datos por medio de cuatro de estos cinco registros de 16 bits, pero nunca con el apuntador de instrucciones.

El registro de banderas

Las *banderas* indican la condición del microprocesador a la vez que controlan su funcionamiento. En la figura 1-8 se ilustran los registros de banderas de todas las versiones de los microprocesadores 8086 a 80486. Se debe tener en cuenta que las banderas son compatibles hacia arriba desde el 8086/8088 hasta el 80486. El 8086-80286 contiene un registro de banderas, FLAG (16 bits) y los 80386-80486 contienen un registro EFLAG (registro de banderas extendido, de 32 bits).

Los bits de bandera cambian después de ejecutar muchas de las instrucciones aritméticas y lógicas. Algunas de las banderas se utilizan para controlar ciertas características del microprocesador. A continuación aparece una lista con cada bit de bandera, con una breve descripción de su función. A medida que se dan a conocer instrucciones en capítulos posteriores, se darán detalles adicionales de los bits indicadores.

C (*acarreo*). Indica un acarreo después de una suma o un "préstamo" después de una resta. La bandera de acarreo también indica condiciones de error en ciertos programas y procedimientos.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
				0	D	I	T	S	Z		A		P		C	8086/8088/80186

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	NT	IOP 1	IOP 0	0	D	I	T	S	Z		A		P		C	80286

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
														VM	RF		NT	IOP 1	IOP 0	O	D	I	T	S	Z		A		P		C	80386/80486DX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
													AC	VM	RF		NT	IOP 1	IOP 0	O	D	I	T	S	Z		A		P		C	80486SX

Nota: Los bits en blanco se reservan para uso futuro y no se necesita definirlos.

FIGURA 1–8 Los registros de banderas de todos los miembros de la familia. Observe que todos son compatibles en forma ascendente.

P (*paridad*). Es un cero para una paridad impar y un 1 para paridad par. La paridad es un conteo de "unos" expresado como un número par o impar. Por ejemplo, si un número contiene 3 bits con uno binario, tiene paridad impar. Si un número contiene cero bits de uno, se considera que tiene paridad par.

A (*acarreo auxiliar*). Tiene un acarreo después de una suma o un "préstamo" después de una resta entre las posiciones de los bits 3 y 4 en el resultado. Este indicador muy especializado se prueba con las instrucciones DAA y DAS para ajustar el valor de AL después de una suma o resta BCD. El microprocesador, no utiliza en otra forma el bit de bandera A.

Z (*cero*). Indica que el resultado de una operación aritmética o lógica es cero. Si Z = 1, el resultado es cero y si Z = 0, el resultado no es cero.

S (*signo*). Indica el signo aritmético del resultado después de una suma o una resta . Si S = 1, la bandera de signo se activa y el resultado es negativo. Si S = 0, la bandera de signo se desactiva y el resultado es positivo. Se debe tener en cuenta que el valor de la posición del bit más significativo se coloca en el bit de signo para cualquier instrucción que afecte las banderas.

T (*trampa*). Cuando se activa la bandera de trampa, se habilita la característica de depuración del microprocesador. Más adelante aparecen mayores detalles de esta característica.

I (*interrupción*). Controla el funcionamiento de la terminal de la entrada de INTR (interrupción). Si I = 1, se habilita la entrada INTR y si I = 0, se deshabilita la entrada INTR. El estado de la bandera I se controla con las instrucciones STI (activar la bandera I) y CLI (desactivar la bandera I).

D (*dirección*). Controla la selección de incremento o decremento de los registros DI o SI durante las instrucciones de cadenas o arreglos. Si D = 1 hay decremento automático en los registros y si D =0 hay incremento. La bandera D se activa con las instrucciones STD (activar dirección) o se borra con CLD (quitar dirección).

O (*sobreflujo*). Es una condición que ocurre cuando se suman o restan números con signo. Un sobreflujo indica que el resultado ha excedido de la capacidad de la máquina. Por ejemplo, si se suma un 7FH (+127) a 01H (+1) el resultado es 80H (-128). Este resultado representa una situación de sobreflujo señalado por la bandera para la suma con signo. Para operaciones sin signo, no se toma en cuenta esta bandera.

IOPL (*nivel de privilegio de entrada/salida*). Se utiliza en el funcionamiento en modo protegido para seleccionar el nivel de privilegio de los dispositivos de E/S. Si el nivel de privilegio actual es de mayor prioridad que el del IOPL, entonces se ejecuta la operación de E/S. Si el nivel del IOPL es menor que el nivel de privilegio actual, ocurre una interrupción y ocasiona que se suspenda la ejecución. Se debe tener en cuenta que un IOPL de 00 es el de mayor prioridad y un IOPL de 11 es el menor prioridad.

NT (*tarea anidada*). Indica que la tarea que está en curso está anidada dentro de otra tarea en el funcionamiento en modo protegido. Esta bandera se activa cuando el software anida la tarea.

RF (*reanudar*). Se emplea con la depuración para controlar la reanudación de la ejecución después de la siguiente instrucción.

VM (*modo virtual*). Selecciona el funcionamiento en modo virtual en un sistema con modo protegido. Un sistema de modo virtual permite múltiples particiones de la memoria con DOS.

AC (*comprobación de alineación*). Si se direcciona a una palabra o palabra doble en una dirección impar (byte) o no múltiplo de cuatro para doble palabra, se activa esta bandera. Sólo el microprocesador 80486SX contiene el bit de comprobación de alineación cuyo empleo primordial es con el coprocesador numérico anexo.

Registros de segmentos

Unos registros adicionales, a los que se da el nombre de *registros de segmentos*, generan direcciones en la memoria junto con otros registros en el microprocesador. Hay 4 o 6 registros de segmentos en las diversas versiones de los microprocesadores 8086-80486. Un registro de segmento funciona de una manera diferente en el modo real, por comparación con el funcionamiento del microprocesador en modo protegido. Más adelante en este capítulo se dan detalles de su función en modos real y protegido. A continuación aparece una lista de cada registro de segmento junto con su función en el sistema:

CS (*código*). El segmento de código es una sección de la memoria que tiene los programas y procedimientos utilizados por los programas. El registro de segmento de código define la dirección inicial de la sección de memoria que tiene el código. En el funcionamiento en modo real, define el inicio de una sección de memoria de 64K bytes y en el modo protegido selecciona un descriptor que describe la dirección inicial y la longitud de la sección de memoria que tiene el código. El segmento de código esta limitado a 64K bytes de longitud en el 8088-80286 y a 4G bytes en el 80386/80486.

DS (*datos*). El segmento de datos es una sección de la memoria que contiene la mayor parte de los datos utilizados por un programa. Se les accesa en el segmento de datos con un desplazamiento o con el contenido de otros registros que tienen la dirección del desplazamiento.

ES (*extra o adicional*). El segmento extra o adicional de datos lo utilizan algunas instrucciones para cadenas.

SS (*pila*). El segmento de pila define la superficie de la memoria utilizada para la pila. La ubicación del punto inicial de entrada a la pila, se determina por el registro apuntador de la pila. El registro BP también direcciona los datos que hay dentro del segmento de pila.

FS y GS. Estos registros de segmento adicionales están disponibles en los microprocesadores 80386 y 80486 a fin de contar con dos segmentos adicionales de memoria para acceso con los programas.

1–5 DIRECCIONAMIENTO DE LA MEMORIA EN MODO REAL

Los microprocesadores 80286-80486 funcionan en el modo real o en el protegido. Los 8086, 8088 y 80186 sólo funcionan en el modo real. En esta sección se detalla el funcionamiento del

microprocesador en el modo real. El funcionamiento en el modo real permite que el microprocesador sólo direccione al primer 1M byte de espacio en la memoria, aunque sea un microprocesador 80486. En los sistemas de funcionamiento MSDOS o PCDOS se supone que el microprocesador funciona en el modo real en todo momento. El funcionamiento en el modo real permite que el software de aplicación escrito para el 8086 u 8088 que sólo contienen 1M byte de memoria, funcionen en los microprocesadores 80286, 80386 y 80486. En todos los casos, cada uno de los microprocesadores empieza a funcionar en modo real en forma implícita (default), siempre que se aplica la corriente o si se restablece el microprocesador.

Segmentos y desplazamientos

Una dirección de segmento y una dirección de desplazamiento, generan una dirección en la memoria en el modo real. Todas las direcciones en la memoria en modo real consisten de un segmento y un desplazamiento. El segmento ubicado en uno de los registros de segmento, define la dirección inicial de cualquier segmento de memoria de 64K bytes. La dirección de desplazamiento selecciona una localidad dentro del segmento de memoria de 64K bytes. En la figura 1-9 se ilustra cómo el esquema de direccionamiento de segmento más desplazamiento selecciona una localidad en la memoria. En esta ilustración se muestra un segmento de memoria que empieza en la localidad 10000H y termina en la 1FFFFH, de 64K bytes de longitud. También se muestra la forma en que un *desplazamiento* de F000H selecciona la localidad 1F000H en el sistema de la memoria. Se verá que la dirección de desplazamiento es la distancia *desde* el inicio del segmento.

El registro de segmento de la figura 1-9, contiene 1000H, con lo que apunta a una dirección inicial 10000H. En el modo real, a cada registro de segmento se le agrega un *0H* en su extremo derecho, para formar una dirección de memoria de 20 bits que le permite colocar el inicio del

FIGURA 1-9 Modo de direccionamiento en modo real con el empleo de una dirección en el segmento más un desplazamiento.

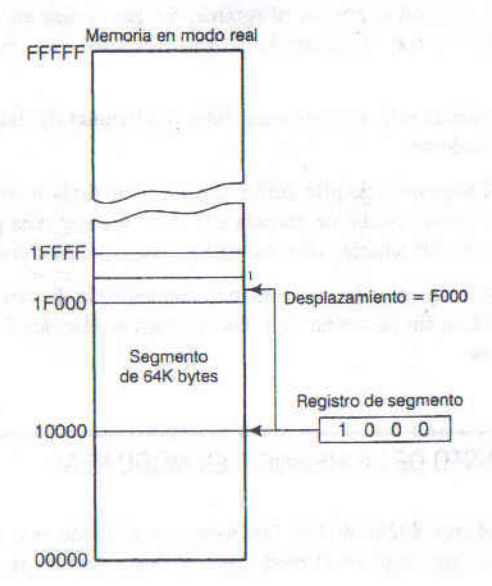


TABLA 1-4 Ejemplo de direcciones de segmentos

Registro de segmento	*Dirección inicial*	*Dirección final*
2000H	20000H	2FFFFH
2100H	21000H	20FFFH
AB00H	AB000H	BAFFFH
1234H	12340H	2233FH

segmento en casi cualquier localidad dentro del primer Mbyte de memoria. Por ejemplo, si un segmento de registro contiene una 1200H, direcciona a un segmento de memoria de 64K bytes que empieza en la localidad 12000H. Asimismo, si un registro de segmento contiene una 1201H direcciona a un segmento de memoria que comienza en la localidad 12010H. Debido al OH, que se agrega en forma interna, los segmentos pueden empezar en cualquier múltiplo de 16 bytes en el sistema de memoria. A menudo, a este límite de 16 bytes se le llama un *párrafo* de memoria.

Debido a que un segmento de memoria en modo real tiene 64K bytes de longitud, una vez que se conoce la dirección inicial, para encontrar la dirección final se agrega una FFFFH a la dirección inicial. Por ejemplo, si un registro de segmento contiene 3000H, la primera dirección en el segmento es 30000H y la última dirección es 30000H + FFFFH o 3FFFFH. En la tabla 1-4 se presentan algunos ejemplos de contenidos de registros de segmento y las direcciones inicial y final de los segmentos de memoria, seleccionadas por cada dirección de segmento.

La dirección del desplazamiento se suma a la del segmento para ubicar una dirección en el segmento. Por ejemplo, si la dirección de segmento es 1000H y la dirección de desplazamiento es 2000H, el microprocesador direcciona la localidad de memoria 12000H. La dirección del segmento y del desplazamiento, a veces, se escribe *1000:2000* para una dirección de segmento de 1000H y un desplazamiento de 2000H.

Registros de segmento y desplazamiento implícitos

El microprocesador tiene un grupo de reglas que se aplican siempre que se direcciona a la memoria. Estas reglas, que se aplican en el modo real o en el protegido, definen la combinación de registro de segmento y de desplazamiento, que se utilizan en ciertos modos de direccionamiento. Por ejemplo, el registro de segmento de código se emplea siempre con el apuntador de instrucciones para direccionar la siguiente instrucción a ejecutar en un programa. Esta combinación es CS:IP o CS:EIP, según sean el microprocesador y el modo de funcionamiento. El registro de segmento de código define el principio de un segmento de código y el apuntador de instrucciones apunta a la siguiente instrucción dentro del segmento de código a ejecutar por el microprocesador. Por ejemplo, si CS = 1400H IP/EIP = 1200H, el microprocesador busca y lee la siguiente instrucción en la localidad 14000H + 1200H o sea 15200H en la memoria.

Otro direccionamiento implícito es el de la pila. Se hace referencia a los datos de la pila por medio del segmento de pila y por localidad de la memoria a la cual direccciona el apuntador de la pila (SP/ESP) o el apuntador de base. Estas combinaciones se refieren como SS:SP (SS:ESP) o SS:BP (SS:EBP). Por ejemplo, si SS = 2000H y BP/EBP = 3000H, el microprocesador direcciona a la localidad 23000H en la memoria del segmento de pila direccionada por el registro BP/EBP. Se debe tener en cuenta que en el modo real, sólo los 16 bits en la extrema derecha del registro extendido direccionan a una localidad dentro del segmento de memoria. Nunca ponga un número mayor que FFFFH en un registro de desplazamiento si el microprocesador funciona en el modo

TABLA 1-5 Segmento y desplazamientos implícitos del 8086-80286

Segmento	*Desplazamiento*
CS	IP
SS	SP o BP
DS	BX, DI, SI o un número de 16 bits
ES	DI para instrucciones para cadenas

real. Si en el modo real se direcciona a una memoria mayor que 100000H (o 10FFEFH si está instalado el sistema HIMEM) ocasionará que el microprocesador interrumpa el programa e indique un error.

En la tabla 1-5 se ilustran otros direccionamientos implícitos para la memoria para microprocesadores 8086-80286. En la tabla 1-6 se muestran los direccionamientos implícitos en los microprocesadores 80386 y 80486. Se debe tener en cuenta que los microprocesadores 80386 y 80486 tienen una selección mucho mayor de las combinaciones de segmento y desplazamiento que los microprocesadores 8086-80286.

Los 8086-80286 permiten tener cuatro segmentos de memoria; el 80386 y el 80486 permiten 6 segmentos de memoria. En la figura 1-10 se ilustra un sistema que contiene 4 segmentos de memoria. Se debe tener en cuenta que los segmentos de memoria pueden tocarse o incluso traslaparse si no se requieren 64K bytes de memoria para un segmento. Piense que los segmentos son como ventanas que se pueden mover en cualquier superficie de la memoria para accesar a datos o código.

Suponga que un programa de aplicación requiere 1000H bytes de memoria para su código, 190H bytes de memoria para sus datos y 200H bytes de memoria para su pila. Esta aplicación no requiere un segmento adicional. Cuando el Dos coloca este programa en la memoria, se carga el TPA en la primera zona disponible encima de los manejadores y otros programas de la TPA. En la figura 1-11 se muestra la forma en que esta aplicación se almacena en el sistema de memoria. Los segmentos muestran una superposición o traslape debido a que la cantidad de datos que hay en ellos no requiere 64K bytes de memoria. La vista lateral de los segmentos muestra con claridad el traslape y la forma en que los segmentos se pueden desplazar a cualquier parte de la memoria. Por fortuna para todos, el DOS calcula y asigna las direcciones en el segmento. Esto se explica en un capítulo más adelante en que se dan detalles del funcionamiento del ensamblador, del BIOS y del DOS para un programa en lenguaje ensamblador.

TABLA 1-6 Segmento y desplazamientos, implícitos del 80386 y 80486

Segmento	*Desplazamiento*
CS	EIP
SS	ESP O EBP
DS	EAX, EBX, ECX, EDX, EDI, ESI, un número de 8 bits o un número de 32 bits
ES	EDI para instrucciones para cadenas
FS	no implícito
GS	no implícito

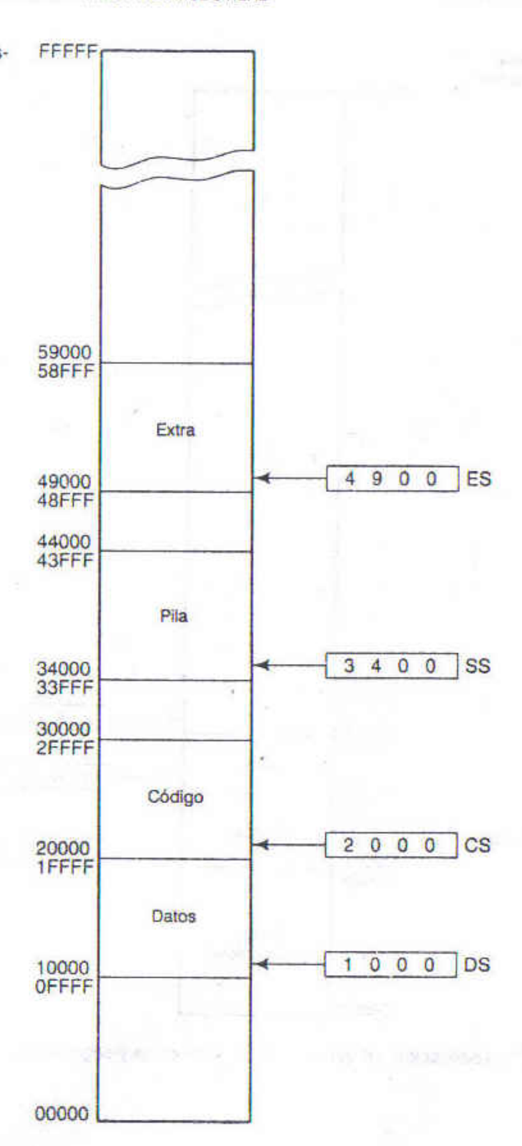


FIGURA 1–10 Un ejemplo de sistema de memoria y se ilustran cuatro segmentos de memoria.

El direccionamiento de segmento y desplazamientó permite la relocalización

El sistema de direccionamiento de segmento y desplazamiento parece ser demasiado complicado. Sí lo es pero también ofrece una ventaja al sistema. El sistema un tanto complicado de direccionamiento de segmento y desplazamiento, permite el cambio de lugar de los programas en

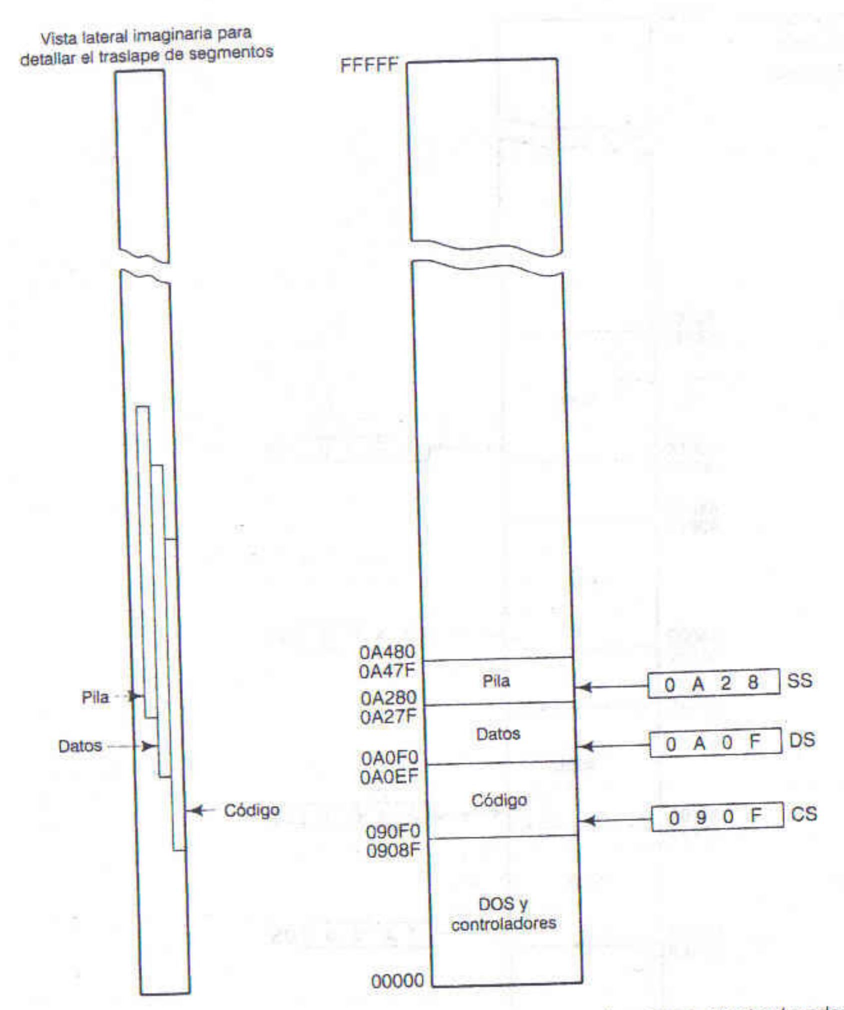


FIGURA 1–11 Mapa de memoria de una aplicación, en que se ilustran segmentos y segmentos traslapados.

el sistema de memoria. Un *programa relocalizable* es el que se puede poner en cualquier zona de la memoria y ejecutarlo sin cambio. Los *datos relocalizables* son los que se pueden colocar en cualquier zona de la memoria y utilizarlos sin ningún cambio en el programa. El sistema de direccionamiento de segmento y desplazamiento permite relocalizar los programas y los datos sin cambiar nada, absolutamente, en un programa o en los datos. Esto es perfecto para utilizarlo en un sistema de computadora para uso general, cuando no todas las máquinas contienen las mismas zonas de memoria. La estructura de la memoria de la computadora personal es diferente entre una máquina y otra, por lo cual requieren software y datos que se puedan relocalizar.

Debido a que a las localidades de memoria se les direcciona dentro de un segmento con una dirección de desplazamiento, el segmento se puede cambiar sin cambiar ninguna de las direcciones de desplazamiento. Para lograrlo, se mueve todo el programa, como bloque, a una nueva zona y sólo se cambia el contenido de los registros de segmento. Si una instrucción está 4 bytes arriba del inicio del segmento, su dirección de desplazamiento es 4. Si se mueve todo el programa a una nueva zona en la memoria, esta dirección de desplazamiento 4 todavía apunta a los 4 bytes que están arriba del inicio del segmento. Lo único que se necesita cambiar es el contenido del registro de segmento para relocalizar el programa en una nueva zona de la memoria. Esta facilidad de relocalización ha hecho que las computadoras personales basadas en microprocesadores de Intel sean muy potentes y muy comunes. Si no se tuviera esta característica, habría que volver a escribir o alterar un programa antes de moverlo. Esto necesitará tiempo adicional o requeriría un programa para cada una de las muchas configuraciones diferentes de los sistemas de computadora.

1–6 DIRECCIONAMIENTO DE LA MEMORIA EN MODO PROTEGIDO

El direccionamiento de la memoria en modo protegido (sólo 80286, 80386 y 80486) permite acceso a los datos y programas ubicados *arriba* del primer Mbyte de memoria. El direccionamiento de esta sección extendida del sistema de memoria (la memoria encima del primer Mbyte de memoria se denomina *memoria extendida o XMS*), requiere un cambio en el sistema de direccionamiento de segmento y desplazamiento, utilizado con el direccionamiento de la memoria en modo real. Cuando se direcciona a los datos y programas en la memoria ampliada, todavía se utilizan la dirección de desplazamiento para accesar la información ubicada dentro del segmento. La dirección del segmento, que se describió en relación con el direccionamiento de la memoria en modo real, ya no está presente en el modo protegido. En lugar de la dirección del segmento, el registro del segmento contiene un *selector* que selecciona un *descriptor*; éste describe la ubicación, longitud y derechos de acceso al segmento de memoria. Debido a que el registro de segmento y la dirección de desplazamiento todavía accesan a la memoria, las instrucciones en el modo protegido se ven iguales que en el modo real. En la práctica, la mayor parte de los programas escritos para funcionar en el modo real, funcionarán sin ningún cambio en el modo protegido. La diferencia entre los modos está en la forma en que el registro de segmento accesa el segmento de memoria.

Selectores y descriptores

El *selector* ubicado en el registro del segmento selecciona a uno de los 8192 descriptores en la tabla de descriptores. El *descriptor* describe la ubicación, longitud y derechos de acceso de un segmento de memoria. El registro de segmento, en forma indirecta, todavía selecciona un segmento de memoria, pero no lo hace en forma directa como en el modo real.

Se utilizan dos tablas de descriptores con los registros de segmento: una contiene descriptores globales y, la otra, contiene descriptores locales. Los *descriptores globales* contienen segmentos que se aplican a todos los programas, mientras que los *descriptores locales* suelen ser exclusivos de una aplicación. Cada tabla de descriptores contiene 8,192 descriptores con lo cual hay disponible en cualquier momento un total de 16,384 descriptores. Debido a que un descriptor describe un segmento de memoria, ello permite describir hasta 16,384 segmentos de memoria para cada aplicación.

Descriptor de 80286

7	00000000	00000000	6
5	Derechos de acceso	Base (B23 - B16)	4
3	Base (B15 - B0)		2
1	Límite (L15 - L0)		0

Descriptor de 80386/80486

7	Base (B31 - B24)	G	D	0	AV	Límite (L19 – L16)	6
5	Derechos de acceso	Base (B23 - B16)					4
3	Base (B15 - B0)						2
1	Límite (L15 - L0)						0

FIGURA 1–12 Los formatos del descriptor para los microprocesadores 80286 y 80386/80486.

En la figura 1-12 se muestra el formato de un descriptor para los microprocesadores 80286 y 80386/80486. Se verá que cada descriptor tiene una longitud de 8 bytes, con lo cual las tablas de descriptores globales y locales tiene, cada una, una longitud máxima de 64K bytes. Los descriptores para el 80286 y los 80386/80486 tienen ligeras diferencias entre sí, pero el descriptor 80286 es compatible en forma ascendente con los microprocesadores 80386 y 80486.

La parte de dirección base del descriptor, se utiliza para indicar la ubicación del inicio del segmento de memoria. En el microprocesador 80286, la dirección de base es una dirección de 24 bits, con lo cual los segmentos pueden empezar en cualquiera de sus 16M bytes de memoria. En el 80386 y 80486 se utiliza una dirección de base de 32 bits, que permite que los segmentos empiecen en cualquiera de las localidades de los 4G bytes de memoria. Se verá cómo la dirección base del descriptor del 80286 es compatible hacia arriba con el descriptor del 80386/80486.

El límite de segmento contiene la última dirección de desplazamiento que se encuentre en un segmento. Por ejemplo, si un segmento empieza en la localidad de memoria F00000H y termina en la ubicación F000FFH, la dirección base es F00000H y el límite es FFH. En el microprocesador 80286, la dirección base es F00000H y el límite es 00FFH. En los microprocesadores 80386/80486, la dirección de base es 00F00000H y el límite es 000FFH. Se verá que el límite para el 80286 es de 16 bits y el límite para 80386/80486 es de 20 bits. El 80286 accesa a segmentos de memoria que tengan longitud entre 1 byte y 64K bytes. El 80386/80486 accesan a segmentos de memoria con longitud desde 1 byte hasta 1M byte o desde 4K bytes hasta 4G bytes.

En el descriptor de 80386/80486 hay una característica que no tiene el descriptor del 80286: el bit G o *bit de granularidad*. Si G = 0, el límite especifica un límite de segmento con longitud entre 1 byte y 1M byte de longitud. Si G = 1, el valor del límite se multiplica por 4K bytes. Si G = 1, el límite puede ser cualquier múltiplo de 4K bytes. Esto permite tener una longitud de segmento entre 4K y 4G bytes, en etapas de 4K bytes. La razón de que esta longitud de segmento sea de 64K bytes en el 80286, es que la dirección de desplazamiento es siempre de 16 bits, mientras que la dirección de desplazamiento cuando se trabaja con 80386/80486 en modo protegido, es de 32 bits. Esta dirección de desplazamiento de 32 bits permite tener longitudes de segmento de 4G bytes y la dirección de desplazamiento de 16 bits, permite tener segmentos de 64K bytes de longitud.

El bit AV en el descriptor de 80386/80486 lo utiliza el sistema operativo e indica que el segmento está disponible (AV = 1) o no disponible (AV = 0). El bit D indica la forma en que las instrucciones del 80386/80486 accesan al registro y a los datos de memoria en el modo protegido. Si D = 0, entonces en el 80386/80486 se supone que las instrucciones son de 16 bits, compatibles

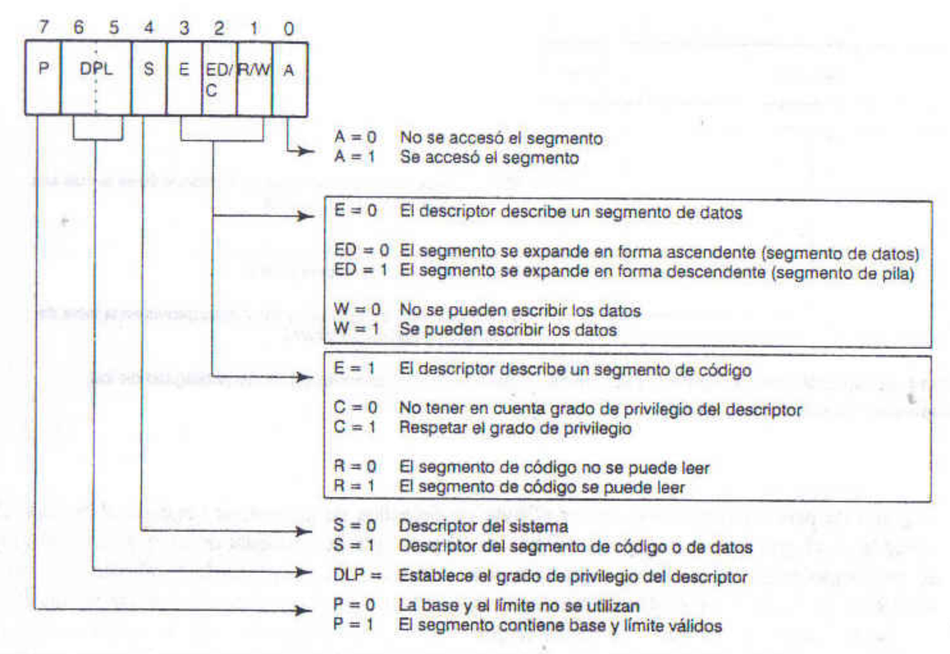


FIGURA 1–13 Los bits de derechos de acceso para los descriptores de 80286, 80386 y 80486.

con los microprocesadores 8086-80286. Esto significa que en las instrucciones emplean direcciones de desplazamiento de 16 bits y registros de 16 bits. Este modo, a menudo, se llama modo de instrucción de 16 bits. Si D = 1, entonces en el 80386/80486 supone que las instrucciones son de 32 bits. El *modo de instrucción de 32 bits*, supone que todos los desplazamientos, así como todos los registros, son de 32 bits. El sistema operativo MSDOS o el PCDOS requieren que las instrucciones se utilicen siempre en el modo de instrucción de 16 bits. En los capítulos 3 y 4 aparecen mayores detalles de estos modos y su aplicación al conjunto de instrucciones.

El byte de derechos de acceso (figura 1-13) controla el acceso al segmento de memoria. Este byte describe cómo funciona el segmento en el sistema. Se puede ver lo completo que es el control del segmento. Si el segmento es de datos, se puede especificar el sentido del crecimiento. Si el segmento crece más allá de su límite, se interrumpe el programa del microprocesador. Es posible especificar también si en un segmento de datos se puede escribir o está protegido contra escritura. El segmento de código se controla en una forma similar y se le puede inhibir la lectura.

Los descriptores los elige en la tabla de descriptores el registro de segmento. En la figura 1-14 se ilustra cómo funciona el registro de segmento en el sistema con modo protegido. El registro de segmento contiene un campo selector de 13 bits, un bit selector en la tabla y un campo del nivel de privilegio solicitado. El selector de 13 bits escoge uno de los 8,192 descriptores en la tabla de descriptores. El bit TI selecciona, ya sea la tabla de descriptores globales (TI = 0) o la tabla de descriptores locales (TI = 1). El nivel de privilegio solicitado (RPL) le solicita el grado de privilegio de acceso a un segmento de la memoria. El grado más alto de privilegio es 00 y el más bajo es 11. Si el grado de privilegio solicitado concuerda o tiene una prioridad mayor que

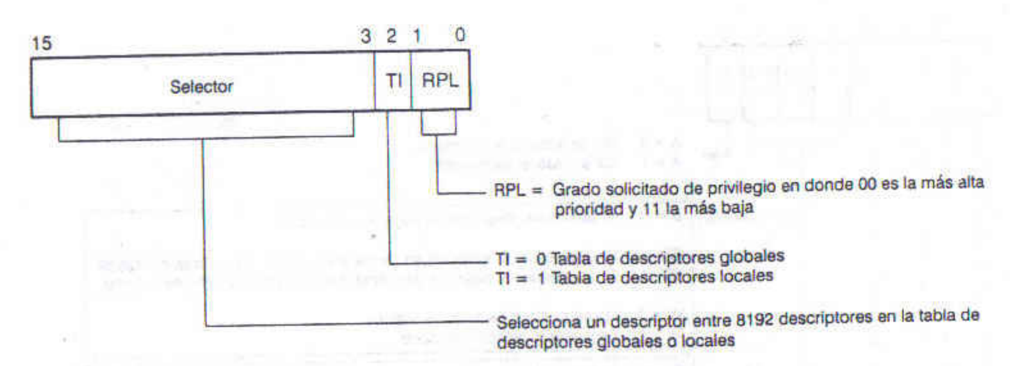


FIGURA 1–14 El contenido de un registro de segmento durante el funcionamiento en modo protegido de los microprocesadores 80286, 80386 u 80486.

el grado de privilegio establecido por el byte de derechos de acceso, se concede el acceso. Por ejemplo, si el grado de privilegio solicitado es 10 y el byte de derechos de acceso, tiene el grado de privilegio en 11, entonces se logra el acceso porque 10 es un grado de privilegio con mayor prioridad que el 11. Los grados de privilegio se emplean en instalaciones con usuarios múltiples.

En la figura 1-15 se ilustra la forma en que el registro de segmento, que contenga un selector, selecciona un descriptor en la tabla de descriptores globales. La entrada en la tabla de descriptores globales selecciona un segmento en el sistema de memoria. En esta ilustración, DS contiene 0008H, que accesa al descriptor número 1 en la tabla de descriptores locales, con el empleo de un nivel de privilegio solicitado de 00. El descriptor número 1 contiene un descriptor 80286 que define la dirección de base como 100000H, con un límite de segmento de 00FFH. Esto significa que un valor de 0008 en DS, hace que el microprocesador utilice las localidades de la memoria 100000H hasta 1000FFH para el segmento de datos con la tabla de descriptor de este ejemplo.

Registros invisibles para el programa

Las tablas de descriptores globales y locales se encuentran en el sistema de memoria. Para poder accesar a la dirección de estas tablas y especificarla, el 80286, 80386 y 80486 contienen registros invisibles para el programa. A los *registros invisibles* para el programa no se les direcciona en forma directa por el software del sistema. En la figura 1-16 se ilustran los registros invisibles para el programa tal como aparecen en los microprocesadores 80286, 80386 y 80486. Estos registros controlan al microprocesador cuando funciona en el modo protegido.

Cada registro de segmento contiene una parte invisible para el programa que se emplea en el modo protegido. Es una memoria caché que se carga con la dirección de base, el límite y los derechos de acceso cada vez que se cambia el número en el registro de segmento. Cuando se carga un nuevo número en un registro de segmento, el microprocesador accesa a una tabla de descriptores y carga el descriptor en la parte de la caché invisible para el programa en el registro de segmento. Se le retiene allí y se emplea para accesar al segmento de memoria hasta que se cambie otra vez el número en el segmento. Esto permite que el microprocesador accese en forma repetida a un segmento de memoria sin consultar la tabla de descriptores para cada acceso.

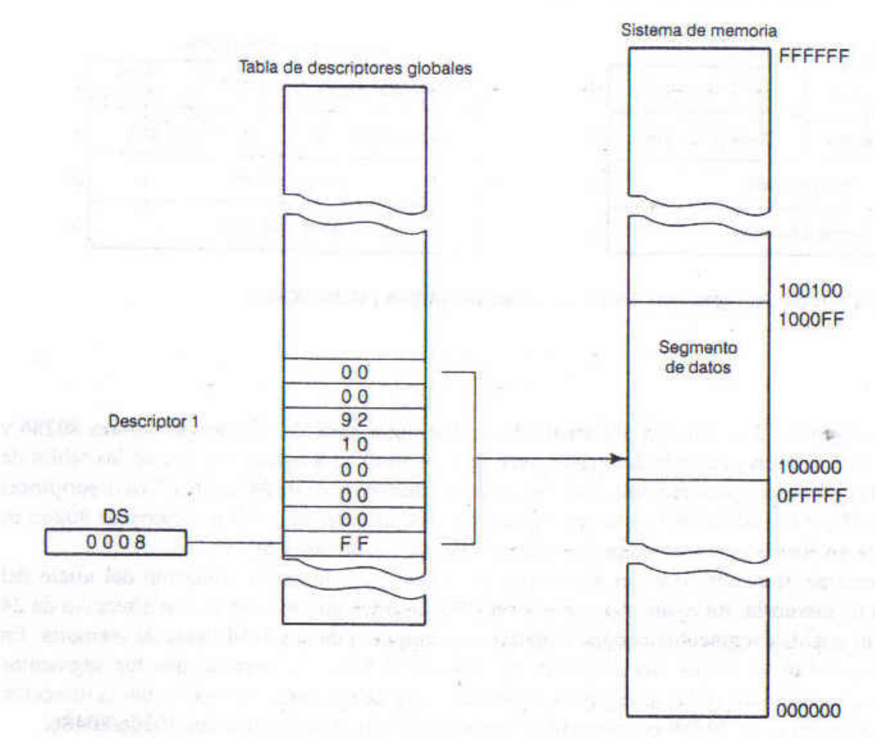


FIGURA 1–15 Empleo del registro DS para seleccionar un descriptor en la tabla de descriptores globales. En este ejemplo, el registro DS accesa a las localidades de memoria 100000H - 10000FFH como segmento de datos.

Los registros GDTR (*registrp de tabla de descriptores globales*) e IDTR (*registro de tabla de descriptores de interrupción*) contienen la dirección base de la tabla de descriptores y su límite. El límite de cada tabla de descriptores es de 16 bits en los microprocesadores 80286, 80386 y 80486, porque la longitud máxima de la tabla es de 64K bytes. Cuando se desea funcionamiento en el modo protegido, se carga la dirección de la tabla de descriptores globales y su límite en el GDTR. Antes de emplear el método protegido hay que inicializar la tabla de descriptores de interrupción y el IDTR. En el capítulo 14 se presentan más detalles del funcionamiento en modo protegido. En el capítulo 14 se incluye una cobertura completa de los microprocesadores 80386 y 80486 y de su funcionamiento en modo protegido. En este momento, es imposible la programación y la descripción adicional de estos registros.

La ubicación de la tabla de descriptores locales se selecciona en la tabla de descriptores globales. Se inicializa a uno de los descriptores globales para direccionar a la tabla de descriptores locales. Para accesar a la tabla de descriptores locales, se carga el LDTR (*registro de tabla de descriptores locales*) con un selector, igual que se carga un selector en un registro de segmento. Este selector accesa a la tabla de descriptores globales y carga la dirección base, el límite y los derechos de acceso de la tabla de descriptores locales en la sección de caché del LTDR.

El *registro de tarea* (TR) accesa a un descriptor que define una tarea. Una tarea, casi siempre, es un procedimiento o un programa de aplicación. El descriptor del procedimiento o del programa

Registros de segmentos — Caché de descriptor

Registros de segmentos	Dirección base	Límite	Acceso
CS			
DS			
ES			
SS			
FS			
GS			

	Dirección base	Límite	Acceso
TR			
LDTR			

Direcciones en tablas de descriptores

	Dirección base	
GDTR		
IDTR		

Invisible para el programa

Notas:

1. El 80286 no contiene FS ni GS ni las partes invisibles para el programa de estos registros.
2. El 80286 contiene una dirección base de 24 bits y un límite de 16 bits.
3. Los 80386 y el 80486 contienen una dirección base de 32 bits y un límite de 20 bits.
4. Los derechos de acceso son 8 bits en el 80286 y 12 en los 80386 y 80486.

FIGURA 1-16 Registro invisible para el programa en los microprocesadores 80286, 80386 y 80486.

de aplicación se almacena en la tabla de descriptores globales en la sección de caché del LTDR, a fin de controlar el acceso por medio de los niveles de privilegio.

1-7 FORMATOS DE DATOS

El éxito en la programación depende también de un entendimiento claro de los formatos de datos. En esta sección se describen los formatos comunes de datos utilizados con la familia de microprocesadores 8086-80486. Los datos se presentan como ASCII, BCD, enteros con signo y sin signo de 8 bits (un byte), enteros con signo y sin signo de 32 bits (doble palabra) y enteros sin signo y números reales largos y cortos (o números con punto decimal, flotante).

Datos ASCII

Los datos en código ASCII (siglas en ingles de *Código Estándar Estadounidense para Intercambio de Información*) (véase tabla 1-7) se suelen utilizar para representar caracteres alfanuméricos en la memoria de un sistema de computadora. El código ASCII es de 7 bits y el octavo bit, que es el más significativo, se emplea para mantener la paridad en algunos sistemas. Si los datos de

TABLA 1–7 El código ASCII

								Segundo								
Primero	*X0*	*X1*	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*XA*	*XB*	*XC*	*XD*	*XE*	*XF*
0X	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1X	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2X	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3X	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4X	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5X	P	Q	R	S	T	U	V	W	X	Y	Z	[	u	]	^	_
6X	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7X	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	░

ASCII se utilizan con una impresora, el bit más significativo es 0 para impresión alfanumérica y un 1 lógico para impresión de gráficas.

En la tabla 1-7 se presentan los caracteres de control de ASCII, junto con una breve descripción de la función de cada código. Si se da entrada a los códigos de control en el teclado, se mantiene oprimida la tecla de control seguida por una @ para un 00, por una a para 01, una b para 02 y así sucesivamente.

En la mayor parte de las computadoras personales hay en el código ASCII un grupo adicional de caracteres llamados caracteres ampliados ASCII. Este grupo de caracteres ampliados (tabla 1-8) siempre se exhibirán en la pantalla de video pero no se pueden imprimir en forma correcta. La única impresora que los imprime correctamente es la IBM pro-printer o una impresora que emule a la pro-printer.

TABLA 1–8 El código ASCII ampliado

								Segundo								
Primero	*X0*	*X1*	*X2*	*X3*	*X4*	*X5*	*X6*	*X7*	*X8*	*X9*	*XA*	*XB*	*XC*	*XD*	*XE*	*XF*
8X																
9X																
AX	á	í	ó	ú	ñ	Ñ	ª	º	¿	⌐	¬	½	¼	¡	«	»
BX	░	▒	▓	│	┤	╡	╢	╖	╕	╣	║	╗	╝	╜	╛	┐
CX	└	┴	┬	├	─	┼	╞	╟	╚	╔	╩	╦	╠	═	╬	╧
DX	╨	╤	╥	╙	╘	╒	╓	╫	╪	┘	┌	█	▄	▌	▐	▀
EX	α	β	Γ	π	Σ	σ	µ	τ	Φ	Θ	Ω	δ	∞	φ	ε	∩
FX	≡	±	≥	≤	⌠	⌡	÷	≈	°	•	·	√	η	²	■	

TABLA 1–9 Datos de BCD empacados y sin empacar

Número	*Empacados*	*Sin empacar*
23	00100011	00000010 00000011
237	00000010 00110111	00000010 00000011 00000111
612	00000110 00010010	00000110 00000001 00000010
1,234	00010010 00110100	00000001 00000010 00000011 00000100

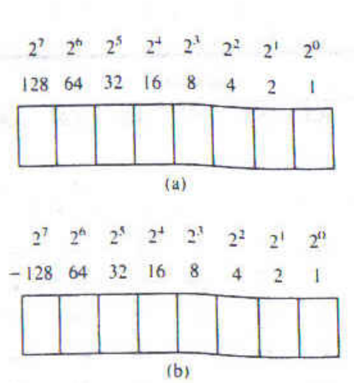
BCD

La información decimal codificada en binario (BCD) se almacena en forma empacada o no empacada en la memoria. La información BCD empacada, almacena dos dígitos BCD por cada byte de memoria; la BCD sin empacar almacena un dígito de BCD por cada byte. Con los datos de BCD los códigos binarios válidos de 4 bits son 0000(0) a 1001(9). En la tabla 1-9 aparecen algunos números decimales en formatos BCD empacado y no empacado.

BYTE

Los datos en Byte se almacenan en dos formas: enteros con signo o sin signo. En la figura 1-17 se ilustran los formatos de enteros con signo y sin signo de ancho de un byte. Se verá que la única diferencia entre las formas con signo y sin signo es la ponderación en la máxima posición de bit hacia la izquierda. En la forma con signo, el bit más a la izquierda es negativo y en la forma sin signo, es positivo o sin signo. Por ejemplo, un 80H es igual a un valor sin signo de 128 y a un valor con signo de –128. Un 81H es igual a un valor sin signo de 129 o a un valor con signo de –127.

FIGURA 1–17 Enteros de 8 bits. (a) Un entero de 8 bits sin signo; (b) un entero de 8 bits con signo.



Aunque los números con signo negativo se representan con el bit de -128 máximo a la izquierda, se almacenan en forma de complemento a dos. Este método para evaluar un número con signo (el bit máximo a la izquierda de −128) es mucho más fácil que el acto de complementar a dos el número para determinar su valor, en especial en donde se emplean calculadoras destinadas a los programadores. Si se desea, se puede hacer uso del complemento a dos para convertir un número negativo a uno positivo a fin de determinar su valor. Para formar el complemento a dos de un número, se invierte cada bit y, luego, se suma un uno al resultado.

Palabra

Una palabra (16 bits) se forma con 2 bytes de datos. El byte menos significativo se almacena siempre en la localidad de memoria que tenga el número más bajo y el byte más significativo, en la más alta. En la figura 1-18(a) se ilustran los valores ponderados de cada bit en una palabra de datos y en la figura 1-18(b) se ilustra cómo se almacena un 1234H en la memoria. La única diferencia entre un número con signo y uno sin signo es el bit de la posición más a la izquierda. En el número con signo su ponderación es negativa y en el número sin signo no es significativa. También en este caso el bit más a la izquierda es el mismo que para los datos de byte, excepto que su valor posicional es diferente.

Dobles palabras

El formato de doble palabra se utiliza para almacenar números de 32 bits (4 bytes), que son el producto después de la multiplicación o el dividendo antes de una división. Las dobles palabras también se utilizan en los 80386 y 80486 para la mayor parte de las operaciones, porque estos microprocesadores más modernos funcionan con datos de 32 bits así como con datos de 8 y 16 bits. En la figura 1-19 se ilustran las ponderaciones binarias y los formatos de almacenamiento en la memoria de una doble palabra. El formato de doble palabra, además de almacenar datos, también se usa para direcciones en la memoria. En la figura 1-20 se ilustra la forma utilizada para

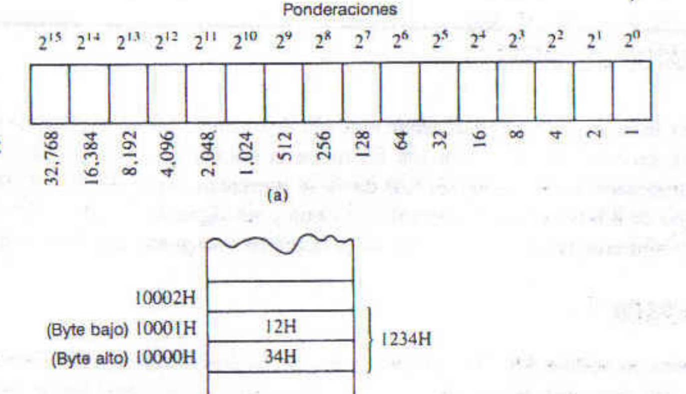


FIGURA 1–18 Enteros de 16 bits. (a) las ponderaciones binarias de cada posición de bit en una palabra de datos de 16 bits. Recuerde que si el número tiene signo, la ponderación del bit más a la izquierda es negativa. (b) Un 1234H almacenado en la memoria, a partir de la ubicación en 10000H.

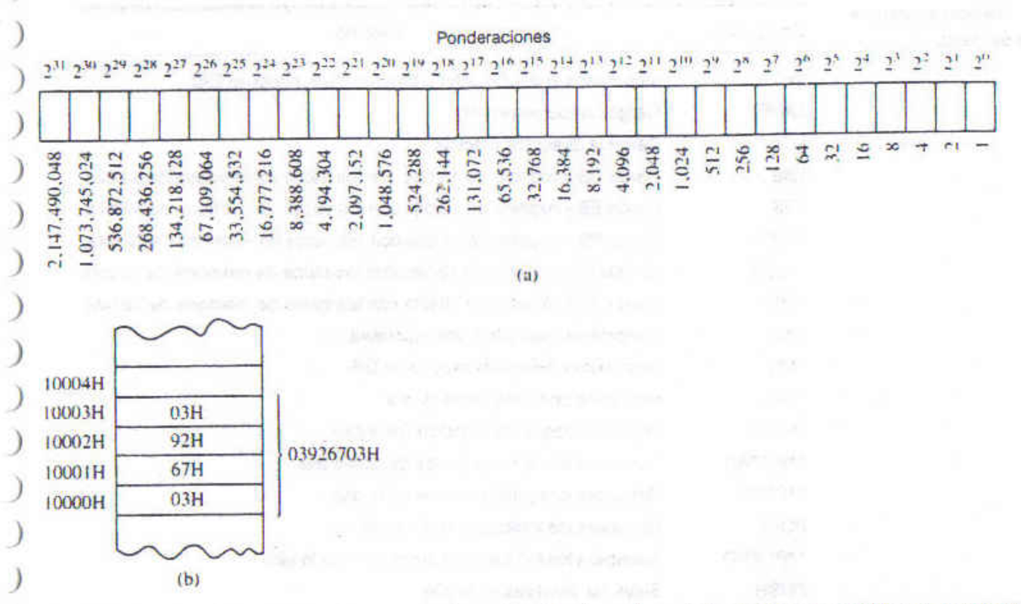


FIGURA 1–19 Enteros de 32 bits. (a) Las ponderaciones binarias de cada posición de bit en una doble palabra de datos de 32 bits. (b) Un 03926703H almacenado en la memoria a partir de la localidad 10000H.

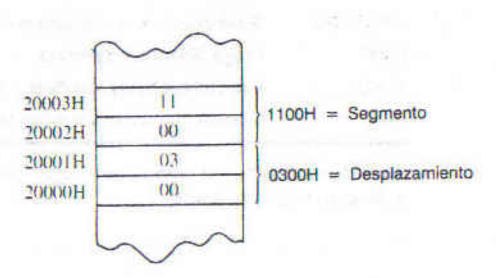


FIGURA 1–20 Un ejemplo de mapa de memoria ilustra cómo se almacena en la memoria la dirección 11300H. El número en el segmento es 1100H y la dirección de desplazamiento es 0300H. Se verá que la dirección del desplazamiento se almacena primero y, luego, la dirección del segmento.

almacenar una dirección de doble palabra. Se verá que su dirección de desplazamiento se almacena en las localidades más bajas de la memoria y que la dirección del segmento se almacena en las más altas.

Números reales

Debido a que la familia 8086/8088 se utiliza con muchos lenguajes de alto nivel y también con algunos sistemas de control muy complejos, a menudo se encuentran números reales. Un *número real* o como también se le llama número con punto decimal flotante, consta de dos partes: una *mantisa* y un *exponente*. En la figura 1-21 se ilustran las formas de 4 y de 8 bits del número real

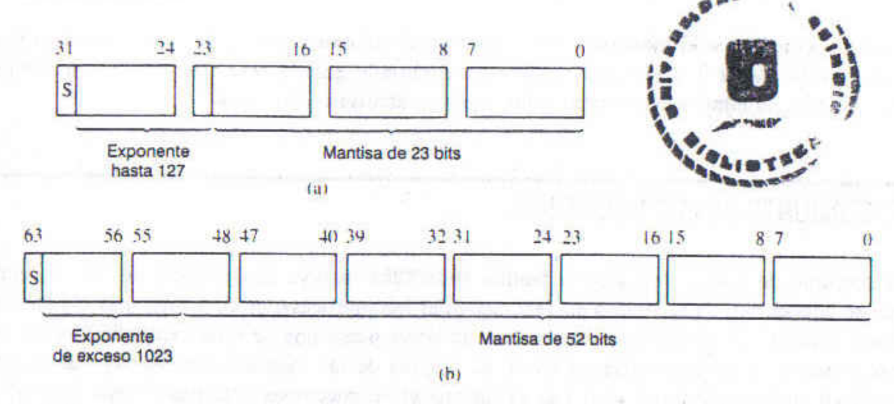


FIGURA 1-21 Almacenamiento de datos de punto flotante o reales. (a) La forma corta de 4 bytes de un número real. (b) La forma larga de 8 bytes de un número real.

que se emplea en el sistema de memoria del microprocesador. Es la misma forma especificada por la norma IEEE-754, versión 10.0. En la figura 1-21(a) se ilustra la forma de precisión sencilla que contiene un exponente de 8 bits, una mantisa (fracción) de 24 bits y un bit de signo.

Un sencillo cálculo aritmético indica que se necesitará un número de 33 bits para almacenar todos estos datos. No es cierto; la mantisa de 24 bits contiene un bit implícito (*oculto*) que permite almacenar la mantisa en 23 bits en vez de 24 bits. El bit oculto es el primer bit del número real normalizado. Cuando se normaliza un número, se le ajusta de modo que su valor sea, cuando menos, de 1 pero menor de 2. Por ejemplo, si se convierte un 12 a binario (1100) y se le normaliza, el resultado es 1.1×2^3. En la tabla 1-10 se ilustra la versión en forma corta de este número y hay otros números de ejemplo convertidos al formato de punto flotante.

El exponente se almacena como *exponente polarizado.* con el número real en forma corta, la polarización es de 127 (7FH) y con el número real de forma larga es de 1,023 (3FFH). En el ejemplo precedente, hay un exponente de 2^3 representado como exponente polarizado de 127 + 3 o 130 (82H) en forma corta y de 1,023 + 3 o 1,026 (402H) en la forma larga. Hay dos excepciones

TABLA 1-10 Notación con números reales en forma corta (precisión sencilla)

Decimal	*Binario*	*Normalizado*	*Signo*	*Exponente polarizado*	*Mantisa*
+12	1100	1.1×2^3	0	10000010	1000000 00000000 00000000
−12	1100	-1.1×2^3	1	10000010	1000000 00000000 00000000
−100	1100100	1.1001×2^6	0	10000101	1001000 00000000 00000000
−1.75	1.11	-1.11×2^0	1	01111111	1100000 00000000 00000000
0.25	0.01	1.0×2^{-2}	0	01111101	0000000 00000000 00000000
0.0	0.0	0.0	0	00000000	0000000 00000000 00000000

a estas reglas: (1) si el número es 0.0, entonces el número real se almacena con un exponente, signo y mantisa de 0, y (2) si el número es demasiado grande para retenerlo en un formato de número real, se almacenan el exponente, signo y mantisa como "unos".

1-8 EL CONJUNTO DE INSTRUCCIONES

El conjunto de instrucciones de la familia 8086/8088 incluye equivalentes de las instrucciones que se encuentran en el 8085 o en casi cualquier microprocesador de 8 bits, más algunas operaciones nuevas. En esta sección se presenta un breve panorama de cada categoría general de instrucciones y de las instrucciones en sí. El empleo de las instrucciones se explica con mayor amplitud en los capítulos 2 al 5. Las categorías de instrucciones descritas en esta sección incluyen: transferencia de datos, aritmética, manipulación de bits, cadenas o arreglos, transferencia de programa y control del procesador.

Transferencia de datos

El conjunto de instrucciones para la familia 8086-80486 incluye instrucciones para transferencia de datos que transfieren bytes, palabras o dobles palabras de datos entre la memoria y los registros así como entre el acumulador y los puertos de E/S. Las transferencias de dobles palabras sólo se pueden hacer en el 80386 y el 80486. En la tabla 1-11 aparecen todas estas instrucciones y se describen en forma breve las características de funcionamiento de cada una.

Aritmética

La familia 8086-80486 puede sumar, restar, multiplicar y dividir datos como bytes, palabras o dobles palabras. Se debe tener en cuenta que las operaciones con dobles palabras sólo se emplean para los microprocesadores 80386 y 80486. El sistema suma y resta con el empleo de datos con signo o sin signo y datos BCD o de ASCII. Multiplica y divide números ASCII con signo o sin signo. En la tabla 1-12 se enumeran las instrucciones aritméticas utilizadas en la familia de microprocesadores 8086-80486.

Manipulación de bits

Las instrucciones para la "manipulación" de bits binarios se utilizan para controlar los datos hasta el nivel de bits en la familia de microprocesadores 8086-80486. Estas instrucciones incluyen operaciones lógicas, corrimientos y rotaciones. En la tabla 1-13 aparece una breve descripción de cada instrucción y de su código simbólico de funcionamiento.

Instrucciones para cadenas

Las instrucciones para cadenas o arreglos se emplean para manipular cadenas de datos en la memoria. Cada cadena consta, ya sea, de bytes o de palabras y tiene hasta 64K bytes de longitud.

TABLA 1–11 Instrucciones para transferencia de datos

Código Op.	*Función*
IN	Mete datos al acumulador desde un dispositivo de E/S
LAHF	Carga banderas en AH
LEA	Carga la dirección efectiva
LDS	Carga DS y registro de 16 bits con los datos de memoria de 32 bits
LES	Carga ES y registro de 16 bits con los datos de memoria de 32 bits
**LFS	Carga FS y registro de 16 bits con los datos de memoria de 32 bits.
**LGS	Carga GS y registro de 16 bits con los datos de memoria de 32 bits
*LSS	Carga SS y registro de 16 bits con los datos de memoria de 32 bits
MOV	Carga byte, palabra o doble palabra
OUT	Saca datos del acumulador a un E/S
POP	Recupera una palabra de la pila
*POPA	Recupera todos los registros de la pila
**POPAD	Recupera todos los registros de doble pila
**POPD	Recupera una palabra doble de la pila
POPF	Recupera los indicadores de la pila
**POPFD	Recupera los indicadores ampliados de la pila
PUSH	Salva las palabras en la pila
*PUSHA	Salva todos los registros en la pila
**PUSHAD	Salva todos los registros de dobles palabras en la pila
**PUSHD	Salva doble palabra en la pila
PUSHF	Salva banderas en la pila
**PUSHFD	Salva banderas ampliadas en la pila
SAHF	Carga AH en las banderas
XCHG	Intercambia bytes, palabras o dobles palabras
XLAT	Emplea AL para entrar a una tabla de conversión

Notas: * = instrucciones nuevas para los 80186 hasta 80486; ** = instrucciones nuevas para 80386 y 80486.

En las instrucciones para la cadena se emplean los registros SI y DI para direccionar los datos y el registro CX para contar el número de bytes o de palabras en que se trabajó. Las instrucciones para las cadenas ocurren una vez salvo que tengan uno de los prefijos REP, REPE/REPZ o REPN/REPNZ. Si una instrucción para cadena tiene uno de esos prefijos, se repite el número de veces contenido en el registro de conteo CX. En la tabla 1-14 se presentan las operaciones para cadena disponibles en la familia de microprocesadores 8086-80486.

Transferencia de programa

Las instrucciones para transferencia de programa incluyen brinco, llamadas (CALL) y para retorno, ya conocidas en el microprocesador 8085. Además se enumeran algunas instrucciones adicio-

TABLA 1-12 Instrucciones aritméticas

Código Op.	*Función*
AAA	Ajuste ASCII para suma
AAD	Ajuste ASCII antes para división
AAM	Ajuste ASCII para multiplicación
AAS	Ajuste ASCII para resta
ADD	Suma datos entre registros o la memoria y otro registro
ADC	Suma datos con la bandera de acarreo
CBW	Convierte byte a palabra
*CDQ	Convierte doble palabra a cuádruple palabra
CMP	Compara los datos
CWD	Convierte palabra a doble palabra
DAA	Ajuste decimal de AL después de una suma de BCD
DAS	Ajuste decimal de AL después de una resta de BCD
DEC	Decrementa
DIV	División sin signo
IDIV	División con signo
IMUL	Multiplicación con signo
INC	Incrementa
*MOVSX	Cargar, ampliar y poner signo a los datos
*MOVZX	Cargar, ampliar datos con ceros
MUL	Multiplicación sin signo
NEG	Cambia el signo (lo vuelve negativo)
SBB	Suma con acarreo
SUB	Resta datos entre los registros y la memoria u otro registro

Nota: * = estas instrucciones sólo aparecen en los microprocesadores 80386 y 80486.

nales para *ciclos* de programas. En la tabla 1-15 se presentan las instrucciones para transferencia de programa.

Control del procesador

Las instrucciones para control del procesador habilitan y deshabilitan las interrupciones, modifican los bits de bandera y sincronizan los eventos externos. En el 80286, 80386 y 80486 controlan el funcionamiento del sistema en el modo protegido. En la tabla 1-16 se presenta una lista de las instrucciones de control del procesador.

TABLA 1–13 Instrucciones para manipulación de bits

Código Op.	*Función*
AND	Y (AND) lógica
*BSF	Rastrear bits hacia el frente
*BSR	Rastrear bits hacia atrás
*BT	Instrucción para prueba de bit
*BTC	Probar bit y complementarlo
*BTR	Probar bit y reactivarlo
*BTS	Probar bit y activarlo
NOT	Invertir (complemento a uno)
OR	O lógica
SAR	Corrimiento aritmético a la derecha
SHL/SAL	Corrimiento a la izquierda
**SHLD	Corrimiento a la izquierda, precisión doble
SHR	Corrimiento lógico a la derecha
**SHRD	Corrimiento lógico a la derecha, doble precisión
RCL	Rotación a la izquierda con acarreo
ROL	Rotación a la izquierda
RCR	Rotación a la izquierda con acarreo
ROR	Rotación a la derecha
TEST	Operación con el AND lógico, pero sólo afectando banderas
XOR	O exclusivo

Nota: * = sólo en los microprocesadores 80386 y 80486; ** = sólo en el microprocesador 80486.

TABLA 1–14 Instrucciones para cadenas

Código Op.	*Función*
CMPS	Comparación entre memoria y memoria
*INS	Meter datos del E/S a la memoria
LODS	Cargar el acumulador
MOVS	Mover de memoria a memoria
*OUTS	Sacar datos de la memoria al espacio de E/S
SCAS	Comparación entre la memoria y el acumulador
STOS	Almacenar en el acumulador

Nota: * = estas instrucciones funcionan en los microprocesadores 80186 hasta 80486.

TABLA 1-16 Instrucciones de control del procesador

Código Op.	*Función*
ARPL	Ajusta grado solicitado de privilegio
CLC	Borrar bandera de acarreo
CLD	Habilitar incremento automático
CLI	Deshabilitar terminal INTR
CMC	Complementa bandera de acarreo
CTS	Borra bandera de conmutación tarea
ESC	Instrucción para el coprocesador
HLT	Alto hasta que se reinicialice o exista interrupción
LAR	Carga derechos de acceso
LGDT	Carga registros de tabla de descriptores globales
LIDT	Carga tabla de registros descriptores de interrupción
LLDT	Carga tabla de registros descriptores locales
LMSW	Carga registro de estado de la máquina (sólo 80286)
LOCK	Controla la terminal $\overline{\text{LOCK}}$ en el 8086 y 8088
LSL	Carga límite de segmento
LTR	Carga registro de tarea
NOP	No operación
SGDT	Almacena tabla de registros de descriptores globales
SIDT	Almacena tabla de registros de descriptores interrupción
SLDT	Almacena tabla de registros de descriptores locales
SMSW	Almacena registro de estado de la máquina (sólo 80286)
STC	Activa bandera de acarreo
STD	Seleccionar modo de decremento automático
STI	Habilitar interrupciones
STR	Almacenar registro de tarea
VERR	Verificar acceso para lectura
VERW	Verificar acceso para escritura
WAIT	Espera a que la terminal $\overline{\text{TEST}}$ = 0

1-9 RESUMEN

1. La duración del microprocesador de 4 bits fue limitada debido a lo reducido de su velocidad, conjunto de instrucciones y tamaño de memoria.
2. El microprocesador de 8 bits resolvió muchos de los problemas que habían surgido, hasta hace poco tiempo, con los microprocesadores de 4 bits. En fechas recientes, los sistemas basados en microprocesadores han empezado a sustituir a las minicomputadoras en muchas aplicaciones que requieren datos de 16 bits, instrucciones adicionales y mucha más memoria de la que tenían los microprocesadores de 8 bits.
3. El microprocesador de 32 bits se utiliza cada vez más debido al aumento en la velocidad de ejecución, un conjunto de instrucciones más amplio y una enorme cantidad de memoria direccionable.

TABLA 1–15 Instrucciones para transferencia de programa

Código Op.	*Función*
*BOUND	Prueba el límite
CALL	Llama a un procedimiento (subrutina)
*ENTER	Entrar al procedimiento
INT	Interrumpir
INT 3	Interrupción tipo 3
INTO	Interrumpir por sobreflujo
IRET	Retornar de una interrupción
*IRETD	Retornar de una interrupción
JA	Brinca si es mayor
JAE	Brinca si es mayor o igual
JB	Brinca si es menor
JBE	Brinca si es menor o igual
JC	Brinca si hay acarreo
JE/JZ	Brinca si es igual o saltar si es cero
JG	Brinca si es mayor que
JGE	Brinca si es mayor o igual
JL	Brinca si es menor que
JLE	Brinca si es menor o igual
JMP	Brinca a otra parte del programa
JNC	Brinca si no hay acarreo
JNE/JNZ	Brinca si no es igual o si no es cero
JNO	Brinca si no hay desbordamiento
JNP	Brinca si no tiene paridad (impar)
JNS	Brinca si no es signo positivo
JO	Brinca si hay desbordamiento
JP	Brinca si hay paridad (par)
JS	Brinca si tiene signo (negativo)
*LEAVE	Abandona el procedimiento
LOOP	Repite ciclo CX veces
*LOOPD	Repite ciclo ECX veces
LOOPE	Formar ciclo mientras sea igual (CX = contador)
*LOOPED	Repite ciclo mientras sea igual (EXC = contador)
LOOPNE	Repite ciclo mientras no sea igual (CX = contador)
*LOOPNED	Repite ciclo mientras no sea igual (ECX = contador)
JCXZ	Brinca si CX es cero
*JECZX	Brinca si ECX es cero
RET	Retorna de un procedimiento (subrutina)

Nota: * = sólo en microprocesadores 80386 y 80486.

4. Un cambio en la organización interna de los microprocesadores de 16 y 32 bits, ha hecho más eficiente el uso de los canales conectados a la memoria. La cola del "prerrecuperador", que lee las instrucciones mientras la memoria está ociosa, permite un uso más eficiente del sistema de memoria. El "prerrecuperador" mantiene la cola llena de modo que el microprocesador ejecute el programa a una velocidad más alta de la que sería posible sin el "prerrecuperador".
5. La memoria lógica es el sistema de memoria que ve el programador; la memoria física es la estructura real de la memoria que puede ver el diseñador de hardware. Los mapas de memoria lógica y física del 8088 son idénticos; los mapas de los otros miembros de la familia son diferentes. La memoria física del 8086, 80186, 80296 y 80386SX está construida con dos bancos de memoria, de un byte, separados, mientras que el 80386DX y el 80486 contienen una memoria física construida con cuatro bancos de memoria de un byte.
6. Los modelos de programación de los microprocesadores 8086, 8088, 80186 y 80286 son idénticos. Los modelos de programación del 80386 y el 80486 son casi idénticos a los de los miembros anteriores de la familia, excepto que la mayor parte de los registros se han ampliado a 32 bits. Todos los miembros de la familia contienen registros de uso general, registros índice y apuntadores, registros de segmentos y un registro de banderas.
7. Los registros de uso general se emplean como cuatro registros de 32 bits (EAX, EBX, ECX y EDX), como cuatro registros de 16 bits (AXBX, CDX y SDCX) o como ocho registros de 8 bits (AH, AL, BH, BL, CH, CL, DH y DL). Se debe tener en cuenta que sólo los 80386 y 80486 contienen registros ampliados de 32 bits.
8. Hay cinco registros de apuntadores y de índices (SP, BP, IP, SI y DI) disponibles para el programador. En los microprocesadores 80386 y 80486 hay también versiones ampliadas de estos cinco registros (ESP, EBP, EIP, ESI y EDI).
9. Los registros de segmentos contienen un número de 16 bits al que se agrega, en el extremo derecho, un 00002 para formar una dirección de 20 bits cuando el microprocesador funciona en el modo real. Esto permite al microprocesador accesar 64K bytes de memoria por segmento. Los cuatro segmentos de memoria disponibles para el programador son código, datos, pila y extra. En los microprocesadores 80386 y 80486, hay dos segmentos adicionales llamados FS y GS.
10. Para formar todas las direcciones de memoria en modo real (dirección efectiva), se emplea un registro de segmento para direccionar un segmento de 64K bytes en la memoria, más un desplazamiento. El desplazamiento es un número de 16 bits que se suma a la dirección del segmento para formar la dirección en la memoria.
11. La palabra de estado tiene banderas similares al 8085 (los 8 bits del extremo derecho) y banderas adicionales. Algunas de las nuevas banderas se emplean para controlar las interrupciones (I), seleccionar la dirección (D) de la característica de incremento e decremento automático en las instrucciones para cadenas, para detectar un sobreflujo (O) y para controlar el modo de funcionamiento instrucción por instrucción (T).
12. La memoria arriba del primer megabyte en los 80286, 80386 y 80486 es accesible siempre que se hace funcionar el microprocesador en el modo protegido. El funcionamiento en modo protegido permite tratar el contenido del registro de segmento como si fuera un selector; éste selecciona un descriptor, en una tabla de descriptores, que describe el segmento de memoria.
13. Los formatos de datos constan de bytes (8 bits), palabras (16 bits) y dobles palabras (32 bits).
14. Los números reales se expresan en formatos corto (32 bits) o largo (64 bits) con la aplicación de la norma IEEE-154. A menudo se llama a la forma corta un número de precisión sencilla de punto flotante y a la forma larga un número de punto flotante de doble precisión.

15. El conjunto de instrucciones del microprocesador incluye instrucciones que permiten transferencia de datos, aritmética, manipulación de bits, operaciones en cadenas, transferencia de programa y control del procesador.

1-10 CUESTIONARIO Y PROBLEMAS

1. ¿Cuáles fueron algunos de los problemas de los primeros microprocesadores de 4 bits?
2. Enumere algunas aplicaciones de los primeros microprocesadores de 4 bits.
3. ¿Qué mejoras en la tecnología de los microprocesadores condujeron a la aparición del microprocesador de 8 bits?
4. Haga una comparación de las velocidades de ejecución de los microprocesadores de 4, 8, 16 y 32 bits.
5. ¿Cuánta memoria direcciona el microprocesador 8086?
6. ¿Cuánta memoria direcciona el microprocesador 80386?
7. Explique por qué el 80486 es más rápido que algunos de los microprocesadores anteriores.
8. ¿Qué es el "paralelismo"? ¿Por qué permite al microprocesador ejecutar el software con más eficiencia?
9. ¿Cuáles son los tres canales conectados con la memoria y la E/S del microprocesador?
10. Los microprocesadores 80386 y 80486 pueden direccionar ______ bytes de memoria.
11. El microprocesador 80286 direcciona ______ bytes de memoria.
12. El 80386 direcciona una memoria que tiene un ancho de ______ bytes.
13. La memoria lógica está numerada del ______ al ______ en el microprocesador 8086.
14. Una palabra requiere ______ bytes de memoria.
15. Una doble palabra requiere ______ bytes de memoria.
16. ¿Cuál es la diferencia entre los mapas de memoria lógica y física del 8088?
17. ¿Cuál es la diferencia entre los mapas de memoria lógica y física del 80486?
18. Un banco de memoria tiene capacidad para almacenar ______ bytes en el sistema de memoria del microprocesador.
19. ¿Qué es la EMS en una computadora personal y dónde se encuentra?
20. El sistema de memoria ampliada (EMS) empieza en la localidad ______ de la memoria.
21. ¿Cuánta memoria se encuentra en la TPA en un sistema de computadora?
22. ¿Cuántos registros de propósito general de 8 bits están disponibles en la familia de microprocesadores 8086/8088? ¿Cómo se llaman?
23. ¿Cuántos registros de propósito general de 16 bits están disponibles en la familia de microprocesadores 8086/8088? ¿Cómo se llaman?
24. ¿Cuántos registros de propósito general de 32 bits están disponibles en el microprocesador 80386? ¿Cómo se llaman?
25. ¿Por qué al registro CX se le llama registro contador?
26. ¿Por que al registro DX se le llama registro de datos?
27. Enumere los cinco registros apuntadores, índices y explique su función normal.
28. Los registros de segmento se utilizan para direccionar un bloque de memoria de 64K bytes en el modo real. ¿Cómo es posible si un registro de segmento sólo tiene 16 bits y la dirección de la memoria tiene 20 bits?
29. ¿Qué registros de segmento se agregaron a los microprocesadores 80386 y 80486?

30. ¿Se pueden traslapar los segmentos de memoria? Si es así, ¿cuál es el número de bytes traslapados que no sea 0?
31. Si IP = 1000H y CS = 2000H, entonces la dirección en modo real de la siguiente instrucción se encuentra en la localidad ______ de la memoria.
32. Si SS = 1234H y SP = 0100H, entonces la dirección actual de la pila es ______.
33. ¿Cuáles son los dos apuntadores que utilizan el registro de segmento de pila para direccionar la memoria?
34. La cadena fuente (SI) se encuentra en el segmento ______ y la cadena destino (DI) se encuentra en el segmento ______ en las instrucciones para cadenas o arreglos.
35. ¿Cuántos de los 16 bits de bandera del 8086 contienen, en realidad, información?
36. Enumere y describa la función de cada uno de los bits de bandera similares al 8085.
37. ¿Cuál es la finalidad del bit de bandera IOPL en el 80386?
38. Explique dónde se utiliza el bit de bandera D y para qué se utiliza.
39. ¿Qué es un sobreflujo?
40. Un byte = ______ bits, una palabra = ______ bits; una doble palabra= ______ bits.
41. Los números con signo y sin signo son bytes, palabras y dobles palabras.) (Indique si es CIERTO o FALSO.)
42. Muestre cómo se almacena un 1234H en una palabra y en una doble palabra, si tanto la palabra como la doble palabra empiezan en una dirección 10000H.
43. Muestre cómo se almacena la dirección 1000:1234 en una doble palabra que empieza en la dirección 04000H.
44. ¿Qué registro mantiene al selector en el modo de sistema protegido?
45. ¿Qué modo de funcionamiento se debe emplear para accesar a la memoria arriba del primer Mbyte en el microprocesador 80386?
46. Convierta los siguientes números a números binarios de 16 bits con signo:
    a. −105
    b. +302
    c. −12
    d. +134
    e. −1003
47. Convierta los siguientes números binarios de 8 bits a valores decimales, con signo y sin signo:
    a. 10000000
    b. 00101011
    c. 11011011
    d. 00111111
    e. 10001111
48. Convierta los siguientes números decimales a números reales de forma corta según la norma IEEE-754:
    a. +10
    b. −11
    c. +101.125
    d. −65.0625
    e. +300.09375

CAPITULO 2

Modos de direccionamiento

INTRODUCCION

El desarrollo de un software eficiente para los 8086-80486 requiere el conocimiento completo de los modos de direccionamiento utilizados por cada instrucción. En este capítulo se emplea la instrucción MOV (mover datos) para describir los modos de direccionamiento de datos. La instrucción MOV transfiere bytes o palabras de datos entre los registros o entre los registros y la memoria en los 8086-80286, y bytes palabras o dobles palabras en los microprocesadores 80386 y 80486. En la descripción de los modos de direccionamiento de la memoria en el programa, se emplean las instrucciones de llamada y de salto que modifican el movimiento del programa.

Los modos de direccionamiento de datos incluyen: registro, inmediato, directo, indirecto por registro, base más índice, relativo por registros y relativo por base más índice, en los microprocesadores 8086-80286. Los microprocesadores 80386 y 80486 incluyen también un modo de índice escalado para direccionar los datos en la memoria. Los modos de direccionamiento de la memoria del programa incluyen: relativo al programa, directo e indirecto. El funcionamiento de la pila de memoria se explica de modo que se puedan comprender las instrucciones PUSH y POP.

OBJETIVOS DEL CAPITULO

1. Explicar el funcionamiento de cada modo de direccionamiento de datos.
2. Utilizar los modos de direccionamiento de datos para formar instrucciones en lenguaje ensamblador y de máquina.
3. Explicar el funcionamiento de cada modo de direccionamiento de la memoria del programa.
4. Utilizar los modos de direccionamiento de la memoria del programa para formar instrucciones en lenguajes ensamblador y de máquina.
5. Seleccionar el modo de direccionamiento adecuado para efectuar una tarea dada.
6. Dar detalles de la diferencia entre el direccionamiento de datos en la memoria al emplear los modos de operación real y protegido en los microprocesadores 80386 y 80486.
7. Describir la secuencia de eventos con los que se cargan y recuperan datos en la pila de memoria.

–1 MODOS DE DIRECCIONAMIENTO DE DATOS

Debido a que la instrucción MOV para los 8086-80486 es sencilla y flexible, proporciona la base para la explicación de los modos de direccionamiento de datos. En la figura 2-1 se ilustra la instrucción MOV y define la dirección del flujo de datos. La fuente está a la derecha y el destino a la izquierda, junto al código de operación MOV. (Un *código de operación* le indica al microprocesador la operación que debe ejecutar.) El sentido del flujo es, al principio, confuso porque se supone que las cosas se mueven naturalmente de izquierda a derecha, mientras que en este caso se mueven de derecha a izquierda. Se verá que en una instrucción *siempre* hay una coma que separa el destino de la fuente.

En la figura 2-1 se ve que la instrucción MOV AX,BX transfiere el contenido de palabras del registro fuente (BX) al registro destino (AX). La fuente *nunca* cambia, pero el destino *normalmente* cambia.[1] Puede ayudar a recordar que una instrucción MOV hace una copia de los datos de fuente y la transfiere al destino.

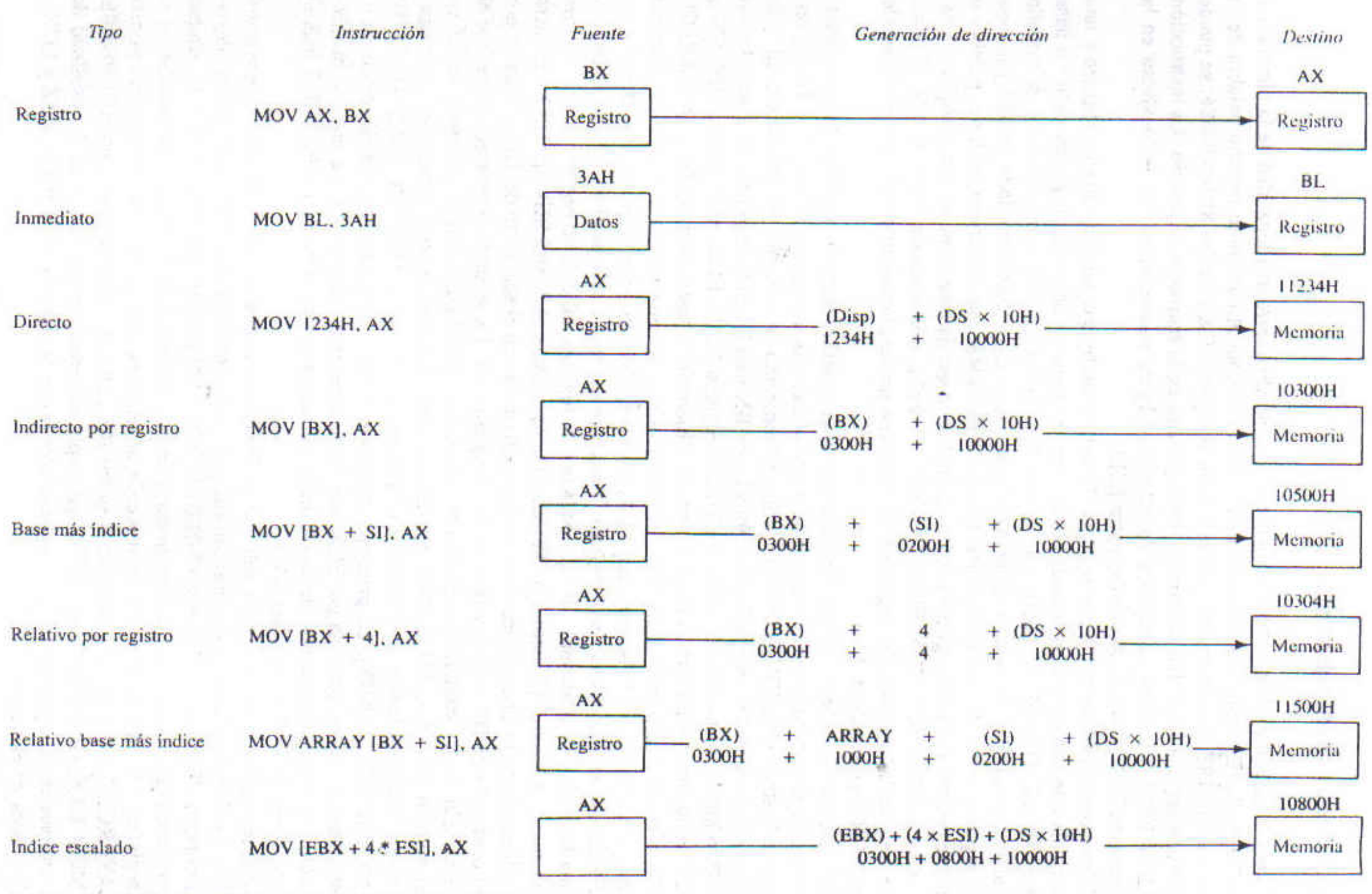
En la figura 2-2 se muestran todas las variantes de los modos de direccionamiento de datos con el empleo de la instrucción MOV. Esta ilustración ayuda a mostrar cómo se formula cada modo de direccionamiento de datos con la instrucción MOV y sirve también de referencia. Se verá que son los mismos modos de direccionamiento de datos de todos los microprocesadores 8086-80486, excepto en el modo de direccionamiento con índice escalado. El direccionamiento con índice escalado sólo se encuentra en los microprocesadores 80386 y 80486; en éstos, los modos de direccionamiento de datos son:

1. *Direccionamiento por registro:* Transfiere un byte o palabra desde el registro fuente o localidad en la memoria, hasta el registro o localidad destino en la memoria. (Ejemplo: la instrucción MOV CX,DX copia el contenido de tamaño de palabra en el registro DX y lo pasa al registro CX.) En el microprocesador 80386/80486, se puede transferir una doble palabra desde el registro de fuente o localidad en la memoria hasta el registro o localidad de memoria de destino. (Ejemplo: la instrucción MOV ECX,EDX copia el contenido de tamaño doble palabra del registro EDX al registro EDX.)
2. *Direccionamiento inmediato:* Transfiere un byte o palabra de datos inmediatos hacia el registro o localidad en la memoria en el destino. (Ejemplo: la instrucción MOV AL,22H copia el 22H de tamaño byte y al registro AL.) En el microprocesador 80386/80486, se puede transferir una doble palabra de datos inmediatos hacia un registro o localidad en la memoria. (Ejemplo: la instrucción MOV EBX,12345678H copia los datos 12345678H que son de tamaño de doble palabra y los transfiere al registro EBX de 32 bytes de ancho).

FIGURA 2–1 La instrucción MOV AB,BX para ilustrar cómo se copia la fuente en el destino.

```
MOV    destino, fuente
MOV    AX, BX
```

[1] La excepción son las instrucciones CMP y TEST que nunca cambian el destino. Estas instrucciones se describen en capítulos posteriores.



Notas: BX = 0300H, SI = 0200H, Matriz = 1000H y DS = 1000H.

FIGURA 2–2

3. *Direccionamiento directo*: Mueve un byte o una palabra entre una localidad de la memoria y un registro. (Ejemplo: instrucción MOV CX,LIST copia el contenido tamaño palabra de la localidad LIST de la memoria y lo coloca en el registro CX.) En los 80386/80486, se puede direccionar una localidad de tamaño doble palabra en la memoria. (Ejemplo: La instrucción MOV ESI,LIST carga un número de 32 bits (4 bytes consecutivos) de la localidad en la memoria LIST y la coloca en el registro ESI.)
4. *Direccionamiento indirecto por registro:* Transfiere un byte o palabra entre un registro y una localidad de memoria direccionada por un registro índice o base. Los registros índice y base son BP, EX, DI y SI. (Ejemplo: La instrucción MOV AX, [BX] copia los datos de tamaño palabra en una dirección del segmento de datos con un desplazamiento dado por BX y lo pasa al registro AX.) En los microprocesadores 80386/80486, se transfiere un byte, palabra o doble palabra entre un registro y una localidad de memoria direccionada por cualquier registro EAX, EBX, ECX, EDX, EBP, EDI o ESI. (Ejemplo: la instrucción MOV AL,[ECX) copia un byte de la dirección del segmento de datos con un desplazamiento dado por el contenido de ECX y lo coloca en AL.)
5. *Direccionamiento base e índice:* Transfiere un byte o palabra entre un registro y la localidad de memoria direccionada por un registro base (BP o BX) más un registro índice (DI o SI). (Ejemplo: la instrucción MOV[BX+DI],CL copia el contenido de tamaño byte del registro CL y lo pasa a la localidad de memoria direccionada por BX más DI en el segmento de datos.) En los 80386/80486, se pueden combinar cualesquier registro (EAX, EBX, ECX, EDX, EBP, EDI o ESI) para generar una dirección en la memoria. (Ejemplo: la instrucción MOV[EAX+EBX],CL copia el contenido de tamaño byte del registro CL y lo coloca en la dirección de memoria direccionada por EAX, la base, más EBX, el índice, ubicados en el segmento de datos.)
6. *Direccionamiento relativo por registro:* Transfiere un byte o una palabra entre un registro y una localidad en la memoria direccionada por un registro índice o un registro base y además un desplazamiento. (Ejemplo: MOV AX,[BX+4) o MOV AX,ARREGLO[BX]. La primera instrucción copia una palabra de datos de una dirección en el segmento de datos, formada por el contenido de BX más 4, y la pone en el registro AX. La segunda instrucción transfiere el contenido de la localidad de memoria de una matriz ARREGLO más el contenido de BX en el registro AX.) Los 80386/80486 pueden emplear cualquier registro listado en direccionamiento base más índice para direccionar la memoria. (Ejemplo: MOV AX, [ECX+4] o MOV AX, ARREGLO [EBX]. La primera instrucción copia una palabra de una dirección en el segmento de datos, formada por ECX más 4 y la pone en el registro AX. La segunda instrucción transfiere el contenido de la localidad de memoria con dirección ARREGLO más el contenido de EBX, al registro AX.)
7. *Direccionamiento relativo base más índice:* Transfiere un byte o una palabra entre un registro y la localidad en la memoria direccionada por un registro base más un índice más un desplazamiento. (Ejemplo: MOV, AX,ARREGLO [BX + DI] o MOV AX, [BX + DI + 4]. Ambas instrucciones copian una palabra de datos de una localidad de memoria y la colocan en el registro AX. En la primera instrucción se emplea una dirección formada al sumar ARRRGLO,BX y DI, y la segunda al sumar BX , DI y 4.) (Un ejemplo para 80386/80486: MOV EAX,ARREGLO[EBX+ECX] que copia el contenido de 32 bits de la localidad de memoria en el segmento de datos a que se accesa con la suma de ARREGLO, EBX y ECX y lo coloca en el registro EAX.)
8. *Direccionamiento de índice escalado:* Está disponible en los microprocesadores 80386 y 80486. El segundo registro de un par de ellos, el índice, se modifica por el factor de escala

2X, 4X u 8X para generar la dirección de la memoria del operando. (Ejemplo: la instrucción MOV AL, [EAX+4*EBX] copia el contenido de tamaño byte en la localidad del segmento de datos en la memoria direccionado por EAX más cuatro veces EBX, y lo pone en el registro AL.) Se utiliza el escalamiento para acceder a palabra (2X), doble palabra (4X) o cuádruple palabre (8X) de un arreglo de datos en la memoria. Se debe tener en cuenta que también hay un factor de escala de 1X, pero por lo general está implícito y no aparece en la instrucción. La instrucción MOV AL,[EBX + ECX] es ejemplo en el cual el factor de escala es uno. Como opción, se podría utilizar MOV AL,[EBX + 1*ECX].

2–2 DIRECCIONAMIENTO POR REGISTROS

Es fácil comprender el direccionamiento por registro una vez que se aprenden los numerosos registros que hay en los 8086-080486. Los 8086-80286 contienen los siguientes registros de 8 bits, utilizados con el direccionamiento por registro AH, AL, BH, BL, CH, CL, DH y DL. También contienen los siguientes registros de 16 bits: AX, BX, CX, DX, SP, BP, SI y DI. En los 80386/80486 los registros, ampliados, de 32 bits son EAX, EBX, ECX, EDX, ESP, EBP, EDI y ESI. En algunas instrucciones MOV y en las PUSH POP se emplean registros de segmento de 16 bits (CS, ES, DS, SS, FS y GS) para direccionamiento por registro. Es importante que para las instrucciones se utilicen registros que sean de tamaño uniforme. *Nunca* mezcle un registro de 8 bits con uno de 16 bits, uno de 8 bits con uno de 32 bits o uno de 16 bits con uno de 32 bits porque no lo permite el conjunto de instrucciones del 8086-80486. Esto es cierto aunque una instrucción MOV AX, AL o una MOV EAX, AL parezca correcta. Por supuesto, *no* son validas las instrucciones MOV AX,AL y MOV EAX,AL porque los registros son de tamaños diferentes.

En la tabla 2-1 se presentan algunas versiones de instrucciones MOV entre registros. Es imposible mostrar todas las variantes por las numerosas combinaciones posibles. Por ejemplo, tan sólo el subconjunto de 8 bits de la instrucción MOV tiene 64 variantes diferentes. El único tipo de instrucción MOV para registro que no se permite, es una instrucción MOV entre un registro de segmento y otro. También se debe tener en cuenta que el registro del segmento de código no

TABLA 2–1 Ejemplos de instrucciones direccionadas a registros

Lenguaje ensamblador	*Operación*
MOV AL,BL	Copia BL en AL
MOV CH,CL	Carga CL en CH
MOV AX,CX	Carga CX en AX
MOV SP,BP	Carga BP en SP
MOV DS,AX	Carga AX en DS
MOV DI,SI	Carga SI en DI
MOV BX,ES	Carga ES en BX
MOV ECX,EBX	Carga EBX en ECX
MOV ESP,EDX	Carga EDX en ESP
MOV ES,DS	No se permite (segmento a segmento)
MOV BL,BX	No se permite (tamaños mixtos)
MOV CS,AX	No se permite (el registro de segmento de código no se puede utilizar como registro de destino)

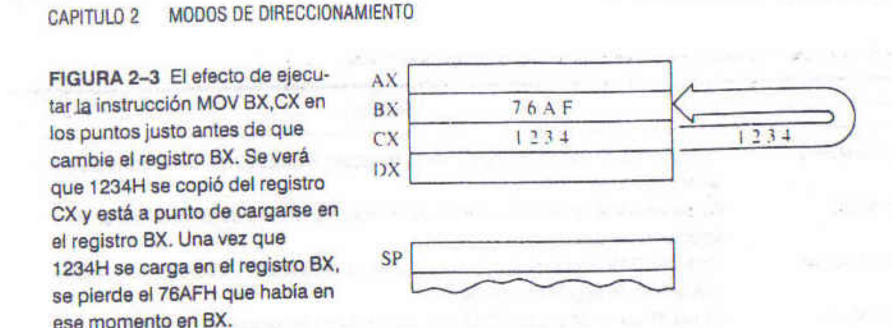


FIGURA 2–3 El efecto de ejecutar la instrucción MOV BX,CX en los puntos justo antes de que cambie el registro BX. Se verá que 1234H se copió del registro CX y está a punto de cargarse en el registro BX. Una vez que 1234H se carga en el registro BX, se pierde el 76AFH que había en ese momento en BX.

se puede cambiar por una instrucción MOV. La razón es que la dirección de la siguiente instrucción se encuentra en el desplazamiento IP/EIP del segmento CS. Si sólo se cambia CS, la dirección de la siguiente instrucción es impredecible.

En la figura 2-3 se ilustra la función de la instrucción MOV BX,CX. Se debe tener en cuenta que el registro fuente no cambia, pero si cambia el destino. Esta instrucción transfiere 1234H del registro CX al registro BX. Esta transferencia de datos borra el contenido anterior (76AFH) del negativo BX, pero el contenido de CX no cambia. El contenido del registro destino o de la localidad de memoria destino cambia con todas las instrucciones excepto las CMP y TEST.

2–3 DIRECCIONAMIENTO INMEDIATO

Otro modo de direccionamiento de datos es el direccionamiento inmediato. El término *inmediato* significa que los datos siguen inmediatamente al código hexadecimal de operación en la memoria. El direccionamiento inmediato actúa en un byte o palabra de datos. En los microprocesadores 80386/80486, el direccionamiento inmediato también actúa en los datos de doble palabra. La instrucción inmediata MOV transfiere una copia de los datos inmediatos a un registro o una localidad en la memoria. En la figura 2-4 se muestra el funcionamiento de una instrucción MOV AX. Esta instrucción copia 3456H de la instrucción, en la memoria y la coloca en el registro AX. Igual que con la instrucción MOV ilustrada en la figura 2-3, los datos fuente borran y sustituyen los datos en el destino.

En lenguaje ensamblador, el símbolo # precede a los datos inmediatos en unos cuantos ensambladores 8086-80486[2]. La instrucción MOV AX,#3456H es un ejemplo de los datos inmediatos. En la mayor parte de los ensambladores no se utiliza el símbolo #, sino que representan los datos inmediatos como en la instrucción MOV AX,3456H. En este libro no se utiliza el símbolo # para los datos inmediatos. En los ensambladores Intel MASM[3] y TASM[4] no se utiliza el símbolo # para los datos inmediatos, pero lo emplea un ensamblador para el sistema de desarrollo lógico HP64000.

El ensamblador simbólico recibe los datos inmediatos en varias formas. La letra H es el sufijo de los datos hexadecimales. Si estos datos empiezan con una letra, se empieza con un 0. Por

[2] Esto ocurre en el ensamblador del sistema HP64100 de desarrollo lógico fabricado por Hewlett-Packard, Inc.
[3] El MASM (macroensamblador) es un programa ensamblador de Microsoft Corporation.
[4] El TASM (turboensamblador) es un programa ensamblador de Borland Corporation.

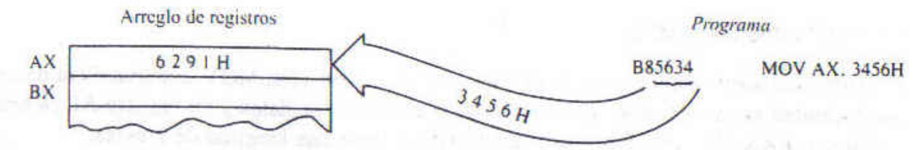


FIGURA 2–4 El efecto de ejecutar una instrucción MOV AX, 3456H. En este caso los datos que siguen al código de operación B8 se cargan desde la memoria hacia el registro AX. Esta operación se ilustra justo en el punto antes de que cambie el registro AX.

ejemplo, para representar un F2H, se utiliza 0F2H en lenguaje ensamblador. Los datos decimales se representan tal cual y no se requieren códigos o ajustes especiales. Un ejemplo es el decimal 100 en la instrucción MOV AL, 100. Uno o más caracteres codificados en ASCII se pueden mostrar en la forma inmediata si los datos ASCII están entre comillas. Un ejemplo es la instrucción MOV BH 'A' la cual mueve a una A (41H) en código ASCII hacia el registro BH. Tenga cuidado de utilizar las comillas (') para encerrar los datos de ASCII. Los datos binarios se representan con el número binario seguido por la letra B o la letra Y. En la tabla 2-2 se presentan varias instrucciones MOV con diferentes datos inmediatos.

2–4 DIRECCIONAMIENTO DIRECTO DE DATOS

En la mayor parte de las instrucciones se puede emplear el modo de direccionamiento directo de datos, el cual se aplica en muchas instrucciones en un programa típico. Hay dos formas básicas de direccionamiento directo de datos: (1) direccionamiento directo que sólo se aplica a una MOV entre una localidad de memoria y AL, AX o EAX, y (2) direccionamiemto por desplazamiento para casi cualquier instrucción en el conjunto de instrucciones de 8086-80486. En cualquier caso, para formar la dirección se suma el desplazamiento a la dirección de segmento de datos implícito o en un segmento alterno.

TABLA 2–2 Ejemplos de direccionamiento inmediato con la instrucción MOV

Lenguaje ensamblador	*Operación*
MOV BL,44	Mueve 44 decimal (2CH) BL
MOV AX,44H	Mueve 44 hexadecimal AX
MOV SI,0	Mueve 0000H hacia SI
MOV CH,100	Mueve 100 (64H) hacia CH
MOV AL,'A'	Mueve ASCII (41H) hacia AL
MOV AX,'AB'	Mueve BA* ASCII (4241H) hacia AX
MOV CL,11001110B	Mueve a un 11001110 binario hacia CL
MOV EBX,12340000H	Mueve 12340000H hacia EBX
MOV ESI,12	Mueve 12 decimal hacia ESI
MOV EAX,100Y	Mueve 100 binario hacia EAX

**Nota*: No es error; los caracteres de ASCII se almacenan como BA, por lo cual se deberá tener cuidado al usar un par de caracteres de ASCII de tamaño palabra.

Direccionamiento directo

El direccionamiento directo sólo se permite con una instrucción MOV que transfiera datos entre una localidad en la memoria, situada dentro del segmento de datos y un registro AL (8 bits), AX (16 bits) o EAX (32 bits). Esta instrucción siempre tiene una longitud de 3 bytes.

La instrucción MOV AL, DATO, tal como la que se emplea en la mayor parte de los ensambladores, transfiere una copia del byte almacenado en la localidad DATA(12345H) de la memoria y los pone en el registro AL. La localidad DATA en la memoria es una *localidad en la memoria simbólica*, en muchos ensambladores se representa esta instrucción como MOV AL,(1234H)[5] La dirección 1234H es una *localidad de memoria absoluta*, la cual no siempre permite todos los programas ensambladores. En la figura 2-5 se ilustra cómo esta instrucción transfiere una copia del contenido de tamaño de byte de la localidad de memoria 1234H, hacia AL. Para formar la dirección efectiva, se suma 1234H (el desplazamiento de la dirección) a 10000H (del segmento de la dirección de los datos) en un sistema que funcione en modo real.

En la tabla 2-3 aparecen tres instrucciones con direccionamiento directo. Estas instrucciones, a menudo aparecen en los programas, por lo cual Intel decidió hacer las instrucciones especiales de 3 bytes de longitud. Todas las demás instrucciones que mueven los datos desde una localidad en la memoria hacia un registro, llamadas instrucciones con direccionamiento por desplazamiento, requieren 4 o más bytes de memoria para almacenarla en un programa.

Direccionamiento por desplazamiento

El direccionamiento por desplazamiento es muy semejante al direccionamiento directo, excepto que la instrucción tiene una longitud de 4 bytes en vez de 3. En los 80386/80486 esta instrucción puede ser de 6 bytes, si se especifica un desplazamiento de 32 bits. Este tipo de direccionamiento directo de datos es mucho más flexible, porque se utiliza en la mayor parte de las instrucciones para 8086-80486.

En la figura 2-6 se muestra el funcionamiento de la instrucción MOV CL,[2000H], la cual funciona en la misma forma que la instrucción MOV AL,(1234H) de la figura 2-5. La diferencia sólo se puede apreciar al examinar las versiones ensambladas de estas dos instrucciones. La instrucción MOV AL, [1234H] tiene 3 bytes de longitud y la MOV CL,[2000H] tiene 4 bytes, como se ilustra en el ejemplo 2-1.

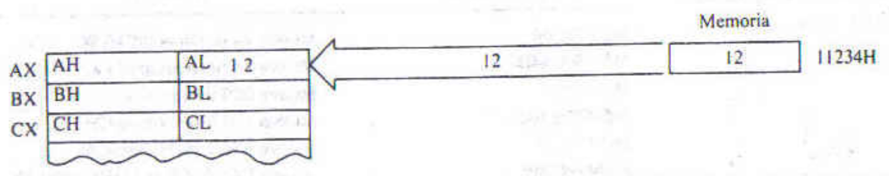


FIGURA 2-5 El efecto de ejecutar la instrucción MOV AL,[1234H] si DS = 1000H. Se ilustra el registro AL después de que los datos (12) han cambiado el contenido del registro AL.

[5] Esta forma se puede utilizar con MASM, pero aparece más a menudo cuando se da, entra o se enlista un programa en DEBUG, una herramienta para depuración incluida en el sistema operativo DOS.

TABLA 2–3 Las cuatro posibles instrucciones direccionadas con AX y AL

Lenguaje ensamblador	*Operación*
MOV AL,NUMERO	Copia en AL el contenido bytes de la dirección NUMERO de la memoria ubicada en el segmento de datos
MOV AX,ALGO	Copia en AX el contenido palabra de la dirección ALGO de la memoria en el segmento de datos
*MOV EAX,AGUA	Copia en EAX el contenido doble palabra de la dirección AGUA en la memoria ubicada en el segmento de datos
MOV NOTAS,AL	Copia AL en la localidad NOTAS de memoria en el segmento de datos
MOV AHI,AX	Copia AX en la localidad de memoria AHI en el segmento de datos
*MOV CASA,EAX	Copia EAX en la localidad CASA del segmento de datos

**Nota:* Los microprocesadores 80386 y 80486 utilizarán en todo momento más de 3 bytes para el movimiento de 32 bits entre EAX y la memoria.

EJEMPLO 2–1

```
0000 A0 1234          MOV  AL,[1234H]
0003 8A 0E 2000       MOV  CL,[2000H]
```

En la tabla 2-4 se presentan algunas formas de direccionamiento directo por desplazamiento, MOV. No se enumeran todas las formas porque hay muchas instrucciones MOV de este tipo. Se debe tener en cuenta que los registros de segmento se pueden almacenar o cargar de la memoria.

Direccionamiento indirecto por registro

El direccionamiento indirecto por registros permite direccionar datos en cualquier localidad de memoria, por medio de cualquiera de los siguientes registros de 8086-80486: BP, BX, DI y SI. Por ejemplo, si el registro BX contiene 1000H y se ejecuta la instrucción MOV AX,[BX] el contenido palabra de la dirección con desplazamiento 1000H en el segmento de datos, se copia en el registro AX. Si el microprocesador funciona en modo real y DS= 0100H, esta instrucción direcciona a la palabra almacenada en los bytes 2000H y 2001H de la memoria se transfieren al registro AX (figura 2-7). Se verá que se mueve el contenido de 2000H hacia AL y el de 2001H a AH. Los símbolos [ ] denotan direccionamiento indirecto en lenguaje ensamblador. Los microprocesadores 80386 y 80486, además de emplear los registros BP, BX, DI y SI, para el direccionamiento indi-

FIGURA 2–6 El efecto de ejecutar la instrucción MOV CL,[2000H) si DS = 1000H. Se ilustra el registro después de que los datos (8A) han cambiado el contenido de CL.

TABLA 2–4 Ejemplos de direccionamiento directo de datos con el empleo de un desplazamiento

Lenguaje ensamblador	*Operación*
MOV CH,CAN	Carga, en el registro CH, el contenido de la localidad CAN de la memoria en el segmento de datos. El ensamblador calcula la dirección de desplazamiento real de CAN
MOV C, [1000H]	*Carga en el registro CH el contenido de la localidad 1000H de la memoria en el segmento de datos
MOV ES,DATO6	Carga en ES el contenido de tamaño palabra de la localidad DATO 6 de la memoria en el segmento de datos
MOV DATO,BP	Se carga a BP en la localidad DATO de la memoria del segmento de datos
MOV NUMERO,SP	Se carga SP en la localidad NUMERO de la memoria del segmento de datos
MOV DATO1,EAX	Se copia EAX en la localidad DATO1 de la memoria del segmento de datos
MOV EDI,SUM1	Carga en el registro EDI el contenido de la localidad SUM1 de la memoria en el segmento de datos

**Nota*: Esta forma de direccionamiento rara vez se usa en la mayor parte de los ensambladores, porque no se direcciona frecuentemente a una dirección numerica de desplazamiento.

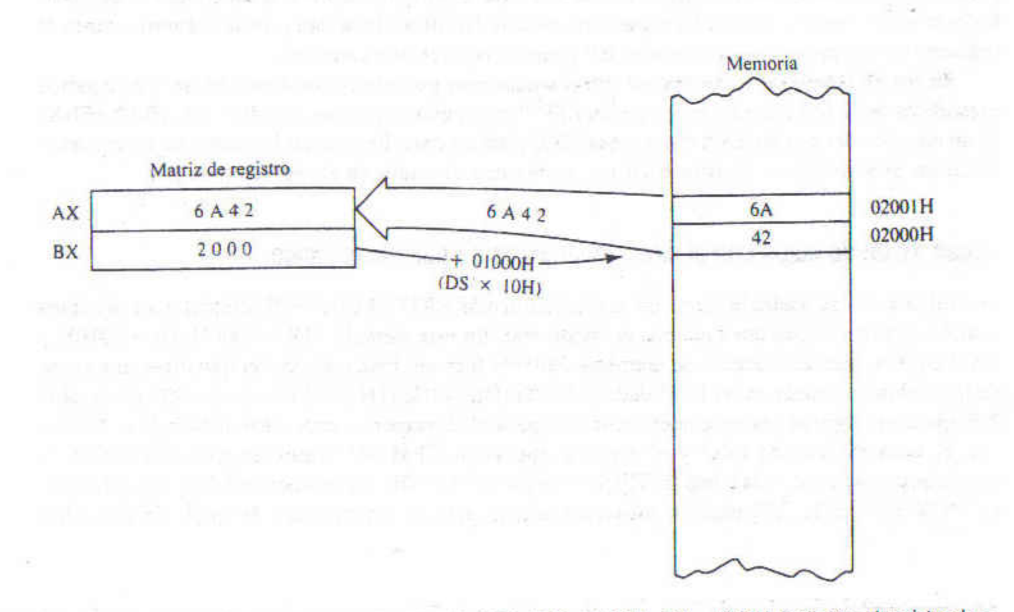


FIGURA 2–7 El efecto de ejecutar la instrucción MOV AX, [BX] si DS= 0100H y BX = 1000H. Se ilustra el registro después de que los datos (6A42H) han cambiado el contenido de AX.

recto permiten también el direccionamiento indirecto por registro con cualquier registro ampliado, excepto el ESP. En la tabla 2-5 aparecen algunas instrucciones típicas que emplean direccionamiento indirecto.

Cuando se utiliza el direccionamiento indirecto por registro o cualquier otro modo de direccionamiento en que se empleen BX, DI o SI para direccionar la memoria, estos registros direccionan los datos en el segmento de datos. Si el registro BP direcciona a la memoria utiliza el segmento de pila. Se consideran implícitos esos segmentos para estos cuatro registros índice y base. Para los 80386 y 80486, el EBP direcciona en forma implícita la memoria, en el segmento de pila mientras que EAX,EBX, ECX, EDX, EDI y ESI direccionan en forma implícita, a la memoria en el segmento de datos. Cuando se utiliza un registro de 32 bits para direccionar a la memoria en modo real, el contenido del registro de 32 bits nunca debe exceder 0000FFFFH. En el modo protegido, se puede emplear cualquier valor en un registro de 32 bits para el direccionamiento indirecto. Por ejemplo, la instrucción para el 80386 y 80486 es MOV EAX, [EBX]. Esta instrucción toma el número de doble palabra almacenado en la dirección de desplazamiento del segmento de datos, señalada por EBX, y lo coloca en EAX.

En algunos casos, el direccionamiento indirecto requiere especificar el tamaño de los datos con directivo BYTE PTR, WORD PTR o DWORD PTR. Estos directivos señalan el tamaño de los datos de la memoria direccionados por el apuntador (PTR). Por ejemplo, es evidente que la instrucción MOV AL,[DI] es una instrucción que transfiere un byte, en tanto la instrucción MOV [DI], 10H es vaga. ¿Direcciona la instrucción MOV [DI],10H a una localidad de tamaño byte, palabra o doble palabra en la memoria? Ni la persona que hace el trabajo ni el ensamblador puede determinar el tamaño. La instrucción MOV BYTE PTR [DI], 10H muestra con claridad que la localidad direccionada por DI es una localidad de tamaño de byte en la memoria. Asimismo, la instrucción MOV DWORD PTR [DI]10H muestra con claridad que la localidad en la memoria es de tamaño de doble palabra. Las directivas BYTE PTR, WORD PTR y DWORD PTR sólo se

TABLA 2–5 Ejemplo de instrucciones con el empleo de direccionamiento indirecto por registro



Lenguaje ensamblador	*Operación*
MOV CX,[BX]	Se carga en el registro CX una palabra de la localidad direccionado por BX en el segmento de datos
MOV [BP],DL	Se carga el byte del registro DL en la localidad de memoria direccionada por BP en el segmento de pila
MOV [DI],BH	Se carga un byte del registro BH en la loacalidad de memoria direccionada por DI en el segmento de datos
MOV [DI],[BX]	No se permiten las transferencias de memoria a memoria excepto con instrucciones de cadena
MOV DI,[DI]	Esta instrucción no se permite, porque el registro utilizado para el direccionamiento indirecto de la memoria no se puede cambiar con la misma instrucción
MOV AL,[EDX]	Se carga en AL un byte de la localidad de memoria en el segmento direccionado por EDX
MOV ECX,[EBX]	Se carga una doble palabra en ECX de la localidad de en el segmento de datos direccionado por EBX

utilizan con instrucciones que direccionan a una localidad en la memoria con un registro apuntador o de índice con datos inmediatos y para unas cuantas instrucciones más que se describen en capítulos posteriores.

El direccionamiento indirecto, a menudo, permite que un programa consulte datos de tablas ubicadas en la memoria del sistema. Por ejemplo, supóngase que se debe hacer una tabla de información que contenga 50 muestras tomadas de un voltímetro digital. En la figura 2-8 se muestran la tabla y el registro BX utilizado para direccionar, secuencialmente, cada localidad de la tabla. Para direccionar la tabla, se necesita cargar la localidad inicial partida de la tabla en el registro BX con una instrucción MOV inmediato. Después de inicializar con la primera la dirección de la tabla, se utiliza el direccionamiento indirecto por registro para almacenar en secuencia las 50 muestras.

EJEMPLO 2–2

```
                   ;instrucciones que leen 50 bytes de datos en PUERTO_DATOS
                   ;y los almacenan en una TABLA.
                   ;
0000 BB 0000 R      MOV   BX BX,OFFSET TABLE     ;dirección TABLA
0003 B9 0032        MOV   CX,50                  ;cargar contador
0006           DE NUEVO:

0006 E4 2A          IN    AL,DATA_PORT           ;leer
0008 88 07          MOV   [BX],AL                ;salvar los datos;
000A 43             INC   BX                     ;direccionar el siguiente
000B E2 F9          LOOP  DE NUEVO
```

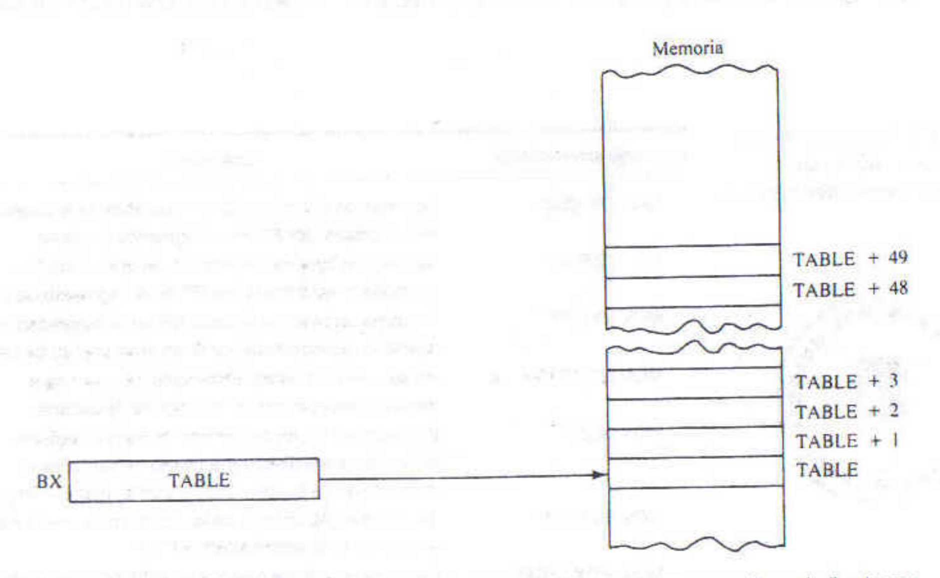


FIGURA 2–8 Un arreglo de datos (TABLA) que contiene 50 bytes que se direccionan en forma indirecta por medio del registro BX.

La secuencia de instrucciones que aparece en el ejemplo 2-2, carga el registro BX con la dirección inicial de la tabla e inicializa a 50 el contador ubicado en el registro CX. El directivo OFFSET le dice al ensamblador que cargue BX con la dirección de desplazamiento de TABLA en la memoria, pero no el contenido de esa TABLA. Por ejemplo, la instrucción MOV BX,DATOS carga el contenido de la localidad DATOS en BX, mientras que la instrucción MOV BX ,OFFSET DATOS copia la dirección de DATOS y la pone en BX. Cuando se utiliza el directivo OFFSET con la instrucción MOV, el ensamblador calcula la dirección y, luego, utiliza una instrucción MOV inmediato para cargar la dirección en el registro de 16 bits especificado.

Una vez inicializados el contador y el apuntador se ejecuta un ciclo-lazo repetir hasta que CX = 0. En este caso, se da entrada (IN) a los datos del voltímetro y se almacenan en la localidad de memoria direccionada en forma indirecta por el registro BX. Después, BX se incrementa (*suma uno*) para apuntar a las siguientes localidades de la tabla y, para concluir, la instrucción LOOP repite el ciclo del programa 50 veces. La instrucción LOOP decrementa (resta uno) el contador CX y si no es cero, LOOP brinca a la localidad otra. Si CX se vuelve cero, no ocurre el brinco y termina el ciclo de instrucciones.

2–5 DIRECCIONAMIENTO BASE MAS INDICE

El direccionamiento base más índice es similar al direccionamiento indirecto, porque direcciona en forma indirecta a los datos de la memoria. En los 8086-80486, se utiliza en este tipo de direccionamiento un registro base (BP o BX) *más* un registro índice (DI o SI) para el direccionamiento indirecto de la memoria. A menudo, el registro base contiene la localidad inicial de un arreglo en la memoria, mientras que el registro índice contiene la posición relativa de un elemento de tamaño byte en el arreglo. Recuérdese que siempre que BP direcciona datos en la memoria, tanto el registro del segmento de pila como el BP generan la dirección efectiva.

En los 80386/80486, este tipo de direccionamiento permite la combinación de dos registros extendidos de 32 bits cualesquiera, excepto ESP. Por ejemplo, la instrucción MOV DL, [EAX+EBX] es un ejemplo del uso de EAX (base) más EBX (índice) para direccionar los datos en el segmento de datos. Si se utiliza el registro EBP, los datos están ubicados en el segmento de pila.

Localización de datos con el direccionamiento de base más índice

En la figura 2-9 se ilustra la forma en que la instrucción MOV DX,[BX+DI] direcciona a los datos cuando el microprocesador funciona en modo real. En este ejemplo, BX = 1000H, DI = 0010H y DS = 0100H, que la dirección de memoria 02010H forman. Esta instrucción transfiere una copia de la palabra contenida en las localidades 02010h (DL) y 02011H (DH) al registro DX. En la tabla 2-6 aparecen algunas instrucciones utilizadas para el direccionamiento base más índice. Nótese que el ensamblador de Intel y el sistema operativo CPM-86[6] requieren que este modo de direccionamiento aparezca como [BX][DI] en lugar de [BX+DI]. La instrucción MOV DX,[BX+DI] es MOV DX, [BX] [DI] para un programa escrito para el ensamblador de Intel. En este libro

[6] CPM-86 y DR DOS versión 6.0 son sistemas operativos producidos por Digital Research Corporation.



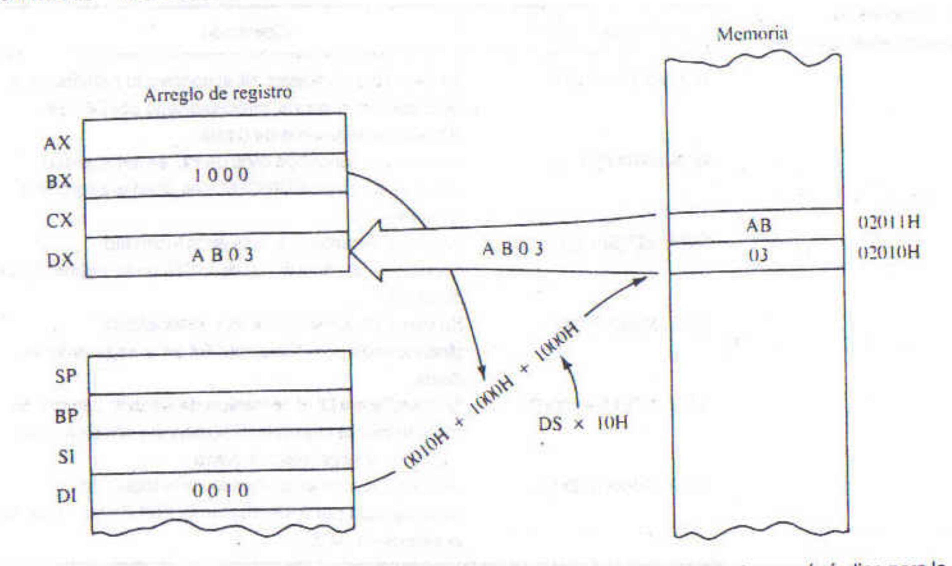


FIGURA 2–9 Un ejemplo de la forma en que funciona el modo de direccionamiento base más índice para la instrucción MOV DX, [BX + DI]. Se verá que la dirección 02010H en la memoria se incluye en esta instrucción porque se suma DS (0100H), BX (1000H) y DI (0010H) para generar esta dirección.

TABLA 2–6 Ejemplos de direccionamiento base más índice

Lenguaje de ensamblador	*Operación*
MOV CX,[BX+DI]	Se carga en el registro CX el contenido palabra de la localidad de memoria direccionada por BX más DI en el segmento de datos
MOV CH,[BP+SI]	Se carga en el registro CH el contenido byte de la localidad de memoria direccionada por BP más SI dentro del segmento de datos
MOV [BX+SI],SP	Se carga el contenido de palabra de SP, en el segmento de datos, en la localidad direccionado por BX más SI
MOV [BP+DI],CX	Se almacena el contenido palabra de CX en la localidad de memoria, direccionada por BP más DI en el segmento de datos
MOV CL,[EDX+EDI]	Se carga en el registro CL el contenido byte de la localidad de memoria, direccionado por EDX más EDI en el segmento de datos
MOV [EAX+EBX],ECX	Se carga el contenido doble palabra de ECX en la localidad de memoria, direccionado por EAX más EBX en el segmento de datos

se utiliza la primera de esas instrucciones en todos los programas de ejemplo, porque es más común.

Localización de datos en un arreglo con direccionamiento base más índice

Uno de los principales usos del modo de direccionamiento base más índice es direccionar los elementos en un arreglo en la memoria. Supóngase que se direcciona a los elementos de un arreglo ubicado en el segmento de datos ubicado en la localidad ARREGLO en la memoria. Para lograrlo, se carga el registro BX (base) con la dirección inicial del arreglo y se carga el registro DI (índice) con el número del elemento. En la figura 2-10 se muestra el empleo de BX y DI para accesar a un elemento en una matriz de datos.

EJEMPLO 2–3

```
     ;empleo del modo de direccionamiento base más índice

0000 BB 0032 R     MOV   BX,OFFSET ARREGLO ;direccionar a ARREGLO
0003 BF 0010       MOV   DI,10H            ;direcciona a elemento 10H
0006 8A 01         MOV   AL,[BX+DI]        ;obtener datos
0008 BF 0020       MOV   DI,20H            ;direcciona a elemento
000B 88 01         MOV   [BX+DI],AL        ;salva los datos
```

Un programa corto, que se presenta en el ejemplo 2-3 sirve para mover al elemento de la localidad 10H de arreglo hacia la 20H. Se verá que el número de elemento del arreglo cargado en el registro DI, selecciona al elemento del arreglo.

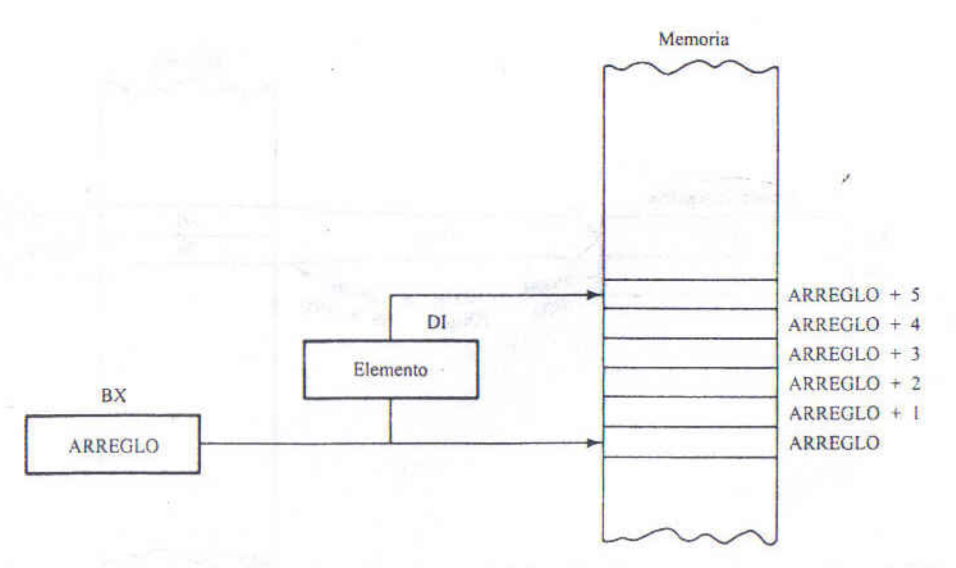


FIGURA 2–10 Un ejemplo del direccionamiento base más índice. En este caso, se accesa a un elemento (DI) de un (ARREGLO) de datos (BX).

2–6 DIRECCIONAMIENTO RELATIVO POR REGISTRO

El direccionamiento relativo por registro es similar al direccionamiento base más índice y al direccionamiento por desplazamiento, que ya se describieron. En el direccionamiento relativo por registro, para direccionar datos en un segmento de memoria se agrega un desplazamiento al contenido de un registro base índice (BP, BX, DI o SI). En la figura 2-11 se muestra el funcionamiento de la instrucción MOV AX,[BX+1000H]. En la figura 2-11 se muestra el funcionamiento de instrucción MOV AX,[BX+1000H]; en este ejemplo, BX = 0100H y DS = 0200H, por lo cual la dirección generada es la suma de DS X 10H, BX, y el desplazamiento de 1000H, que es 03100H. Recuérdese que BX, DI o SI direccionan al segmento de datos y BP direcciona al segmento de pila. En los 80386 y 80486, el desplazamiento puede ser un número de 32 bits y el registro puede ser cualquiera de 32 bits, excepto el ESP. Recuérdese que el tamaño de un segmento en modo real es 64K bytes de longitud. En la tabla 2-7 se presentan algunas instrucciones en las que se utiliza el direccionamiento relativo por registro.

Un desplazamiento es un número que se suma al registro dentro de los corchetes [ ] , como en la instrucción MOV AL, [DI+2] o que se resta del registro como en la instrucción MOVAL,[SI-1]. Un desplazamiento también es una dirección de desplazamiento delante de los [ ] como en MOV AL,DATO[DI]. Ambas formas de desplazamiento pueden aparecer al mismo tiempo como en la instrucción MOV AL,DATO[DI + 3]. En todos los casos, ambas formas de desplazamiento se suman al registro base o base e índice dentro de los corchetes [ ]. En los microprocesadores 8086-80286, el valor del desplazamiento está limitado a un número con signo de 16 bits o sea ± 32K y en los 80386-80486 se permite un desplazamiento de 32 bits.

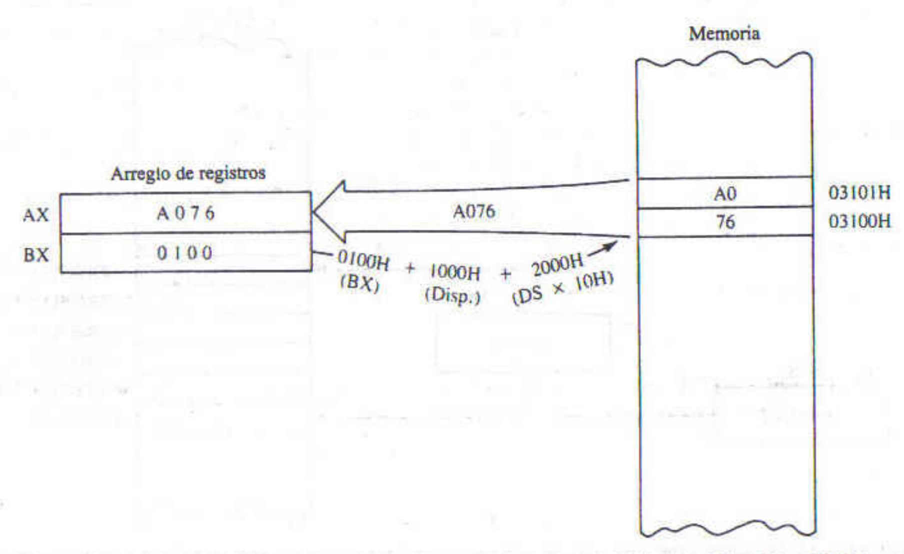


FIGURA 2–11 El efecto de ejecutar la instrucción MOV AX,[BX + 1000H) si BX = 0100H y DS = 0200H.

TABLA 2–7 Ejemplos de direccionamiento relativo por registro

Lenguaje ensamblador	*Operación*
MOV AX,DI+100H]	Se transfiere al registro AX el contenido palabra de la localidad de memoria, direcccionado por DI más 100H en el segmento de datos
MOV ARRAY[SI],BL	Se carga el contenido byte de BL en la localidad direccionado por ARREGLO más SI en el segmento de datos
MOV LIST[SI+2],CL	Se carga el contenido byte en la localidad direccionada por la suma de LISTA en el segmento de datos, SI y 2
MOV DI,SETS[BX]	Se carga DI con el contenido de localidad direccionada por CONJ más BX en el segmento de datos
MOV DI,[EAX+100H]	Se transfiere a DI el contenido de tamaño palabra de la localidad de memoria direccionado por EAX más 100H, en el segmento de datos
MOV ARRAY[EBX],AL	Se carga AL con el contenido de la localidad direccionado por ARREGLO más EBX de memoria, en el segmento de datos

Direccionamiento de un arreglo datos con direccionamiento relativo por registro

Es posible direccionar datos de un arreglo con direccionamiento relativo por registro, en una forma muy parecida a la del direccionamiento base más índice. En la figura 2-12 se ilustra el direccionamiento relativo por registro mediante el mismo ejemplo que utilizamos para el registro base más índice. Esto demuestra cómo el desplazamiento de ARREGLO se suma al registro índice DI para generar una referencia a un elemento del arreglo.

En el ejemplo 2-4, se muestra la forma en que este nuevo modo de direccionamiento puede transferir el contenido del elemento 10H del arreglo al 20H del arreglo. Se mostrará la semejanza entre este ejemplo y el ejemplo 2-3. La diferencia principal es que en el ejemplo 2-4 no se utiliza el registro BX para direccionar el área ARREGLO de la memoria, sino que en vez de ello, se utiliza ARREGLO como desplazamiento para lograr el mismo resultado.

EJEMPLO 2–4

```
          ;empleo de direccionamiento relativo por registro

0000 BF 0010         MOV   DI, 10H             ;direcciona elemento 10H
0003 8A 85 0300 R    MOV   AL, ARREGLO [ DI]   ;obtener datos
0007 BF 0020         MOV   DI, 20H             ;dirección elemento 20H
000A 88 85 0300 R    MOV   ARREGLO [ DI] ,AL   ;salva los datos
```

2–7 DIRECCIONAMIENTO RELATIVO BASE MAS INDICE

El direccionamiento relativo base más índice es similar al direccionamiento base más índice, pero además suma un desplazamiento utilizando un registro base y un registro índice para formar

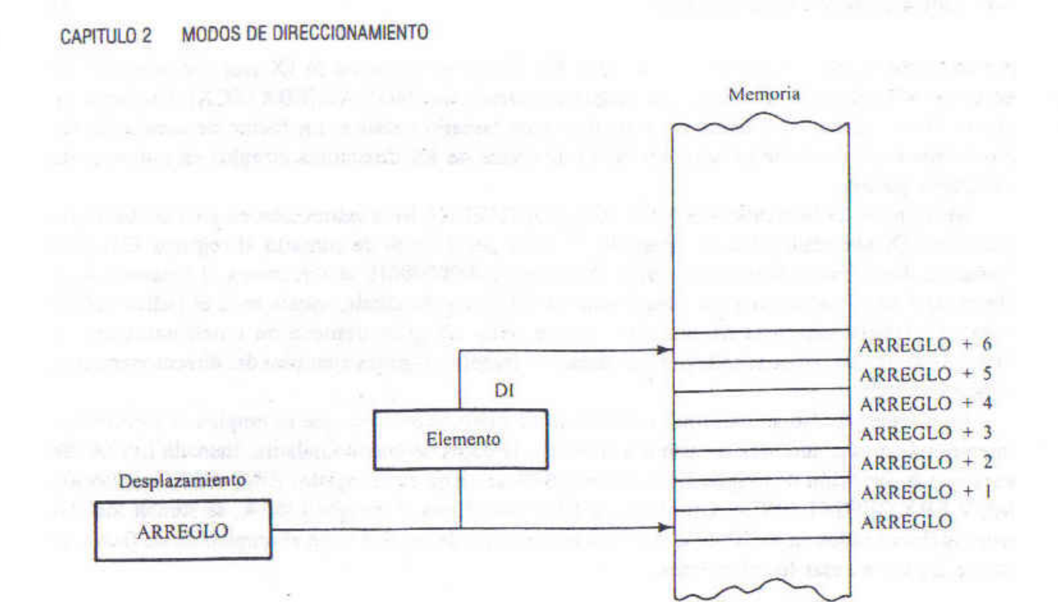


FIGURA 2-12 Direccionamiento relativo por registro utilizado para direccionar a un elemento del arreglo (ARRAY). El desplazamiento direcciona el comienzo de la matriz y el contenido de DI (el registro) selecciona un elemento de la matriz.

una dirección de memoria. Este tipo de direccionamiento, a menudo direcciona un arreglo bidimensional en la memoria.

Direccionamiento de datos relativos base más índice

El direccionamiento relativo base más índice es el modo de direccionamiento menos utilizado. La figura 2-13 muestra la forma en que se hace referencia a los datos si la instrucción ejecutada por el microprocesador es MOV AX,[BX+SI+100H]. El desplazamiento 100H se suma a BX y SI para formar la dirección del desplazamiento dentro del segmento de datos. Los registros BX = 0020H, SI = 0010H y DS = 1000H, con lo cual la dirección efectiva para esta instrucción es 10130H, o sea la suma de estos registros más un desplazamiento de 100H. Este modo de direccionamiento es en extremo complejo para emplearse con frecuencia en un programa. En la tabla 2-8 aparecen algunas instrucciones típicas en que se emplea el direccionamiento relativo base más índice. Se verá que con los microprocesadores 80386 y 80486 la dirección efectiva se genera por la suma de dos registros de 32 bits más un desplazamiento de 32 bits.

Direccionamiento de arreglos con direccionamiento relativo base más índice

Supóngase que hay en la memoria un archivo de muchas clasificaciones y que cada una contiene muchos elementos. El desplazamiento direcciona el archivo, el registro base direcciona una clasificación y el registro índice direcciona un elemento de una clasificación. En la figura 2-14 se ilustra esta forma tan compleja de direccionamiento.

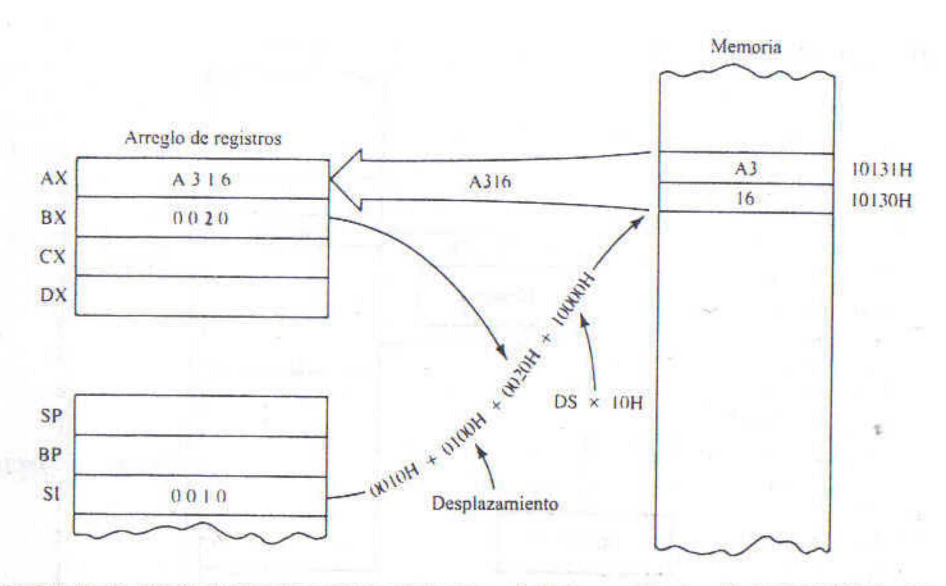


FIGURA 2–13 Un ejemplo de direccionamiento relativo base más índice con la instrucción MOV,100H[BX + SI]. Esta instrucción carga datos desde la memoria al registro AX. La dirección de la memoria es la suma de DS× 10H, 100H, BX y SI.

TABLA 2–8 Ejemplo de instrucciones de direccionamiento relativo base más índice

Lenguaje ensamblador	*Operación*
MOV DH,[BX+DI+20H]	Se carga DH con el contenido de la localidad direccionado por la suma de BX, DI y 20H en el segmento de datos
MOV AX,ARCHIVO[BX+DI]	Se carga AX con el contenido de la localidad direccionado por la suma de ARCH, BX y DI en el segmento de datos
MOV LISTA[BP+DI],CL	Se carga DL en la localidad direccionado por la suma de LISTA, BP y DI en el segmento de pila
MOV LISTA[BP+SI+4],DH	Se carga DH en la localidad del segmento de pila direccionada por la suma de LISTA, BP, SI y 4
MOV AL,ARCHIVO[EBX+ECX+2]	Se carga AL desde la localidad de memoria en el segmento de datos direccionado por la suma de ARCH, EBX, ECX y 2

El ejemplo 2-5 proporciona un programa que copia el elemento 0 del registro A al elemento 2 del registro C utilizando el modo de direccionamiento relativo más índice.

EJEMPLO 2–5

```
        ;empleo del modo de direccionamiento base más índice

0009 BB 0364 R        MOV   BX,OFFSET RECA    ;direcciona registro A
0003 BF 000           MOV   DI,0              ;direcciona elemento 0
```

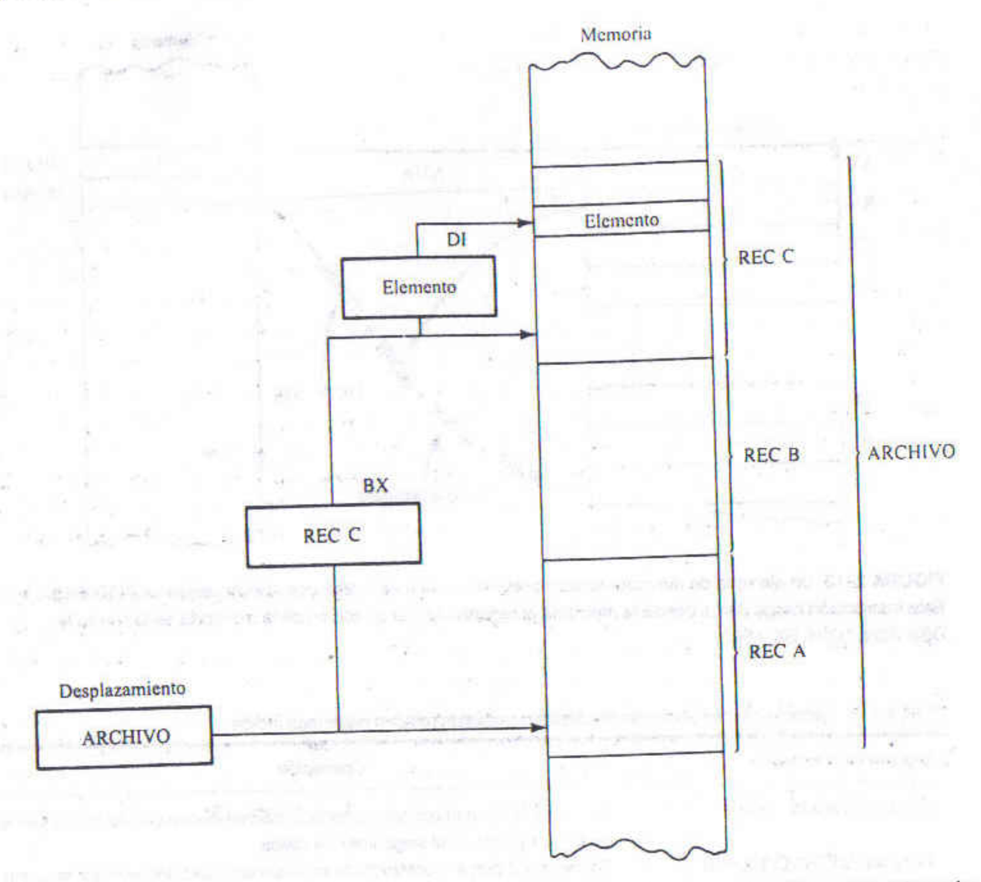


FIGURA 2-14 Direccionamiento relativo base más índice utilizado para direccionar un archivo que contiene clasificaciones múltiples y cada uno de éstos contiene varios elementos.

```
                                                  ;direcciona registro A
                                                  ;direcciona elemento 0
0006 8A 81 04F4 R    MOV  AL,FILE[BX+DI]          ;obtener datos
000A BB 042C R       MOV  BX,OFFSET RECC          ;direcciona registro C
000D BF 0002         MOV  DI,2                    ;direcciona elemento 2
0010 88 81 04F4 R    MOV  ARCHIVO [BX+DI],AL      ;salva los datos
```

2-8 DIRECCIONAMIENTO INDICE ESCALADO

El direccionamiento índice escalado es el último modo de direccionamiento de datos que se va a describir. Este modo de direccionamiento de datos es exclusivo de los microprocesadores 80386 y 80486. En el direccionamiento índice escalado se emplean dos registros de 32 bits (un registro base y un registro índice) para accesar a la memoria. El segundo registro (índice) se multiplica

por un factor de escala; éste es 1X, 2X, 4X u 8X. Un factor de escala de 1X está implícito y no se necesita incluirlo en la instrucción en lenguaje ensamblador MOV AL,[EBX+ECX]. Un factor de escala de 2X direcciona arreglos en memoria de tamaño palabra; un factor de escala de 4X direcciona a los de doble palabra; un factor de escala de 8X direcciona arreglos en memoria de cuádruple palabra.

Un ejemplo es la instrucción MOV AX,[EDI+2*ECX]. Esta instrucción emplea un factor de escala de 2X que multiplica el contenido de ECX por 2 antes de sumarlo al registro EDI para formar la dirección en la memoria. Si ECX contiene 00000000H, se direcciona al elemento 0; al elemento 1 de la memoria y así sucesivamente. El factor de escala, escala en 2 el índice (ECX) para un arreglo memoria de tamaño palabra. Hay un gran número de combinaciones de direccionamientos índice escalado. En la tabla 2-9 aparecen algunos ejemplos del direccionamiento índice escalado.

En el ejemplo 2-6 se muestra la secuencia de instrucciones en que se emplea el direccionamiento índice escalado para accesar a una matriz de datos de tamaño palabra, llamada LISTA. Se verá que la dirección de desplazamiento de LISTA se carga en el registro EBX con la instrucción MOV EBX,OFFSET LISTA. Una vez que EBX direcciona el arreglo LISTA, se suman los elementos (localizados en ECX) de 2, 4 y 7 de este arreglo de palabras con el empleo de un factor de escala 2 para accesar los elementos.

EJEMPLO 2-6

```
                          ;secuencia de instrucciones para sumar elementos de LISTA tamaño de palabra
                          ;
0000 66| BB 00000000      MOV  EBX,0
0006 BB 0000 R            MOV  BX,OFFSET LISTA      ;direcciona el arreglo LISTA

0009 66| B9 00000002      MOV  ECX,2                ;direcciona elemento 2
000F 67& 8B 04 4B         MOV  AX,[EBX+2*ECX]       ;obtener elemento 2
```

TABLA 2-9 Ejemplos de instrucciones en que se utiliza direccionamiento por índice escalado

Lenguaje ensamblador	*Operación*
MOV EAX,[EBX+4*ECX]	Se carga el registro EAX desde la localidad de memoria, en el segmento de datos direccionado por la suma de EBX más 4 veces ECX
MOV [EAX+2*EBX],CX	Se carga el registro CX en la localidad de memoria en el segmento de datos direccionado por EAX más 2 veces EBX
MOV AX,[EBP+2*EDI+100H]	Se carga el registro AX con el contenido de la localidad de memoria en el segmento de pila, direccionado por EBP más dos veces EDI más 100H
MOV LIST[EAX+2*EBX+10H],DX	Se carga el registro DX en la localidad de memoria en el segmento de datos direccionado por LISTA más EAX más 2 veces EBX más 10H

dirección numérica real. Cuando se utiliza una etiqueta con las instrucciones CALL o JMP, la mayor parte de los ensambladores seleccionan la mejor forma posible de direccionamiento del programa.

Direccionamiento relativo de memoria de programa

El direccionamiento relativo a la memoria de programa no está disponible en muchos microprocesadores anteriores, pero sí lo está en los 8086-080486. El término *relativo* significa "relación al apuntador de instrucciones (IP)". Por ejemplo, si una instrucción JMP brinca los 2 siguientes bytes de memoria, la dirección en relación con el puntero de apuntador de las instrucciones es un 2 que se suma al apuntador. Así se produce la dirección de la siguiente instrucción del programa. En la figura 2-16 se ilustra un ejemplo de la instrucción JMP relativo. Se verá que la instrucción JMP es de un byte, con un desplazamiento de 1 o de 2 bytes que se suma al apuntador de instrucciones. En los brincos cortos se utiliza un desplazamiento de un byte; en los brincos y llamadas cercanos se utiliza un desplazamiento de 2 bytes. Ambos tipos se consideran como saltos intrasegmento. (Un salto o brinco *intrasegmento* es dentro del segmento de código presente.) En los microprocesadores 80386 y 80486, el desplazamiento también puede tener un valor de 32 bits, lo cual permite emplear en ellos el direccionamiento relativo para saltar a cualquier localidad del segmento de código de 4G bytes.

Las instrucciones JMP y CALL relativas contienen un desplazamiento con signo de 8 bits o de 16 bits, que permite una referencia hacia delante o hacia atrás en la memoria. (Los 80386 y 80486 pueden tener un desplazamiento de 32 bits.) Todos los ensambladores calculan en forma automática la distancia del desplazamiento y seleccionan la forma correcta de 1, 2 o 4 bytes. Si la distancia es muy grande para un desplazamiento de 2 bytes en microprocesadores 8086-80286, en algunos ensambladores se emplea el salto directo. Un desplazamiento de 8 bits (*corto*) tiene un valor entre +127 y -128, mientras que un desplazamiento de 16 bits (*cercano*) tiene un valor entre +32K. En los 80386 y 80486, un desplazamiento de 32 bits permite un intervalo de ±2G bytes.

Direccionamiento indirecto de memoria de programa

Los 8086-080486 permiten cierto número de formas de direccionamiento indirecto de la memoria del programa para la instrucción JMP y CALL. En la tabla 2-10 se presentan algunas instrucciones aceptables para brinco indirecto al programa, que pueden utilizar cualquier registro de 16 bits (AX, BX, CX, DX, SP, BP, DI o SI), cualquier registro relativo ([BP] [BX], [DI] o [SI]) y cualquier registro relativo con un desplazamiento. En los 80386/80486, un registro extendido también puede contener la dirección o una dirección indirecta para las instruccciones JMP o CALL relativos. Por ejemplo, en los 80386/80486 se puede emplear una JMP EAX para brincar a la localidad direccionada por el registro EAX.

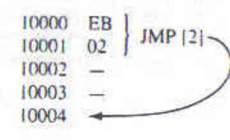


FIGURA 2-16 Una instrucción JMP [2] que brincará sobre los dos bytes siguientes en el programa. En este ejemplo, el programa continúa en la localidad 10004H.

TABLA 2–10 Ejemplos de direccionamiento indirecto del programa

Lenguaje ensamblador	*Operación*
JMP AX	Brinca a la localidad direccionada por AX en el segmento de código actual
JMP CX	Brinca a la localidad direccionada por CX en el segmento de código actual
JMP [BX]	Brinca a la localidad, en el segmento de código actual, la almacenada en la dirección localidad del segmento de datos más BX
JMP [DI+2]	Brinca a la localidad en el segmento de código actual, almacenada en la dirección localidad del segmento de datos más DI + 2
JMP TABLA[BX]	Brinca a la localidad en el segmento del código actual direccionada por TABLA más BX
JMP ECX	Brinca a la localidad en el segmento de código actual direccionada por ECX

Si un registro de 16 bits contiene la dirección de una instrucción JMP, el brinco es cercano. Por ejemplo, si el registro BX contiene una 1000H y se ejecuta una instrucción JMP BX, el microprocesador brinca a la dirección con desplazamiento 1000H en el segmento de código que está en uso.

Si la dirección está contenida en un registro relativo, se considera que el brinco es indirecto. Por ejemplo, un JMP [BX] significa un brinco a la localidad de memoria del segmento de datos con la dirección de desplazamiento contenida en BX. En esta dirección está un número de 16 bits que se emplea como dirección de desplazamiento en el brinco intrasegmentos.

En la figura 2-17 se muestra una tabla de saltos que se almacena empezando en la localidad de memoria TABLA. Esta tabla de brincos se menciona en el programa corto del ejemplo 2-7. En este ejemplo, el registro BX se carga con un 4, de modo que cuando se combina en la instrucción JMP TABLA [BX] la dirección efectiva es el contenido de la segunda entrada en la tabla de salto.

EJEMPLO 2–7

```
                    ;empleo del direccionamiento indirecto para un brinco
                    ;
0000 BB 0004             MOV  BX,4             ;direccionar LOC2
0003 FF A7 23A1 R        JMP  TABLE[BX]        ;saltar hasta LOC2
```

```
TABLA   DW   LOC0  |  Direcciones de cuatro
        DW   LOC1  |  programas diferentes.
        DW   LOC2  |  (Cada dirección es un
        DW   LOC3  |  desplazamiento de 2 bytes.)
```

FIGURA 2–17 Una tabla de brincos que se utiliza para permitir que el programa seleccione diferentes direcciones de salto para diferentes valores en el registro BX.

2–10 DIRECCIONAMIENTO DE LA PILA DE MEMORIA

La pila de memoria es una parte importante del sistema de memoria de todos los microprocesadores. Contiene los datos en forma temporal y almacena las direcciones de retorno para los procedimientos o subrutinas. La pila de memoria es del tipo *LIFO* (último en entrar, primero en salir) en los microprocesadores 8086-80486. Los datos se colocan en la pila con una instrucción PUSH y se recuperan con una instrucción POP. La instrucción CALL aprovecha la pila para "salvar" la dirección de retorno del procedimiento, y la RET para "recuperar" de la pila la dirección de retorno del procedimiento.

La pila de memoria se mantiene con dos registros: el *apuntador de pila* (SP o ESP) y el *segmento de pila* (SS) Siempre que se coloca (PUSH) una palabra en la pila [figura 2-18(a)], los

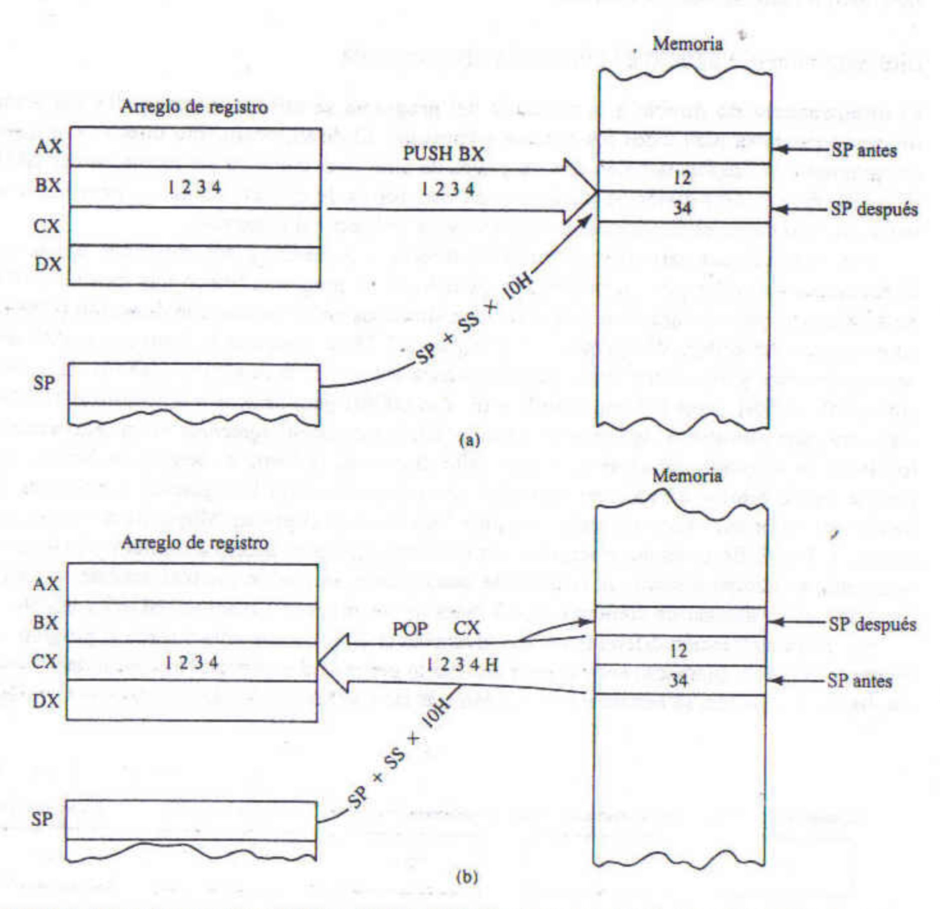


FIGURA 2–18 Las instrucciones PUSH y POP. (a) PUSH BX coloca el contenido de registro BX en la pila direccionada por SP + SS × 10H. (b) POP CX recupera datos de la pila en la localidad direccionada por SP + SS × 10H y coloca los datos del registro CX.

```
0013 B1 04          MOV  CL,4              ;direcciona elemento 4
0015 67& 03 04 4B   ADD  AX,[EBX+2*ECX]    ;suma elemento 4

0019 B1 07          MOV  CL,7              ;direcciona elemento 7
001B 67& 03 04 4B   ADD  AX,[EBX+2*ECX]    ;suma elemento 7
```

2–9 MODOS DE DIRECCIONAMIENTO DE MEMORIA DE PROGRAMA

Los modos de direccionamiento de memoria de programa, utilizados con las instrucciones JMP (Salto) y CALL(Llamada) tienen tres configuraciones distintas: directa, relativa e indirecta. En esta sección se describen estas tres formas de direccionamiento, con el empleo de la instrucción JMP para ilustrar su funcionamiento.

Direccionamiento directo a la memoria del programa

El direccionamiento directo a la memoria del programa se utilizó en casi todos los primeros microprocesadores para todos los brincos y llamadas. El direccionamiento directo a la memoria del programa se emplea también con lenguajes de alto nivel como en las instrucciones GOTO y GOSUB. En los 8086-80486 se puede utilizar esta forma de direccionamiento, pero no lo hacen tan a menudo como el direccionamiento relativo e indirecto al programa.

Las instrucciones para direccionamiento directo a la memoria del programa, almacenan la dirección con el código de operación. Por ejemplo, si un programa brinca a la localidad 10000H de la memoria para la siguiente instrucción, se almacena en la memoria la dirección (10000H) a continuación del código de operación. En la figura 2-15 se muestran la instrucción JMP directa *entre segmentos* y los cuatro bytes requeridos para almacenar la dirección 10000H. Una instrucción JMP 10000H carga CS con 1000H e IP con 0000H para brincar a la localidad 10000H de memoria, para ejecutar la siguiente instrucción. Un brinco *entre segmentos* es un salto a cualquier localidad de memoria del sistema. A este salto directo se le llama a menudo un brinco *lejano*, porque puede brincar a cualquier localidad de la memoria para la siguiente instrucción. En el modo real, el brinco largo accesa a cualquier localidad en el primer Mbyte de memoria porque cambia CS e IP. En el modo protegido, con el brinco lejano se accesa a un nuevo descriptor del segmento de código tomado de la tabla de descriptores, lo cual le permite accesar a cualquier localidad en el sistema de memoria de 4G bytes de los microprocesadores 80386 y 80486.

La única instrucción diferente en que se emplea el direccionamiento directo al programa es la instrucción CALL (llamada) entresegmentos. Por lo general, el nombre de una localidad de memoria, llamado *etiqueta*, se refiere a la localidad a la cual se hace la llamada o se brinca en lugar la

Código de operación	Desplazamiento, bajo	Desplazamiento, alto	Segmento, bajo	Segmento, alto
EA	00	00	00	10

FIGURA 2–15 Instrucción en lenguaje de máquina de 5 bytes para una instrucción JMP (10000H). El código de operación va seguido por la dirección del desplazamiento (0000H) y por la dirección (1000H) del segmento de código.

8 bits de orden alto se colocan en la localidad direccionada por SP-1. Los 8 bits de orden bajo se colocan en la localidad direccionada por SP-2. Luego, se decrementa en 2 el SP de modo que la siguiente palabra se almacene en la localidad de la pila de memoria inmediatamente disponible. El registro SP/ESP siempre apunta hacia un área de la memoria ubicada dentro del segmento de pila. El registro SP/ESP se suma a SS x 10H para formar la dirección de la pila de memoria en el modo real. Al funcionar en modo protegido, el registro SS tiene un selector que accesa a un descriptor para la dirección base del segmento de pila.

Siempre que se recuperan datos (palabra) de la pila [figura 2-18(b)] se recuperan los 8 bits de orden bajo de la localidad direccionada por SP. Los 8 bits de orden alto se recuperan de la localidad direccionada por SP + 1. Luego se incrementa en 2 el registro SP. En la tabla 2-11 aparecen algunas de las instrucciones PUSH y POP disponibles en los microprocesadores 8086-80486. Se debe tener en cuenta que PUSH y POP siempre almacenan o recuperan *palabras*, nunca bytes en los microprocesadores 8086-80286. Los 80386/80486 permiten transferir palabras o dobles palabras hacia y desde la pila. Los datos se pueden colocar dentro de la pila desde cualquier registro o registro de segmento de 16 bits y en los 80386/80486 con cualquier registro ampliado de 32 bits. Los datos se pueden recuperar de la pila hacia cualquier registro o registro de segmento de 16 bits, excepto CS. La razón por la que no se puedan recuperar datos de la pila y para CS, es que con ello sólo se cambia parte de la dirección de la siguiente instrucción.

TABLA 2–11 Ejemplo de instrucciones PUSH y POP

Lenguaje ensamblador	*Operación*
POPF	Recupera una palabra de la pila y la carga en el registro de banderas
POPFD	Recupera una doble palabra de la pila y la carga en el registro EFLAGS
PUSHF	Salva una copia de la palabra del registro de banderas en la pila
PUSHFD	Salva una copia de EFLAGS en la pila
PUSH AX	Almacena una copia de AX en la pila
POP BX	Recupera una palabra de la pila y la pone en BX
PUSH DS	Salva una copia de DS en la pila
PUSH 1234H	Salva un 1234H en la pila
POP CS	Instrucción ilegal
PUSH [BX]	Salva el contenido palabra en la localidad de memoria direccionada por BX en el segmento de datos en la pila
PUSHA	Salva el contenido de los registros AX, CX, DX, BX,SP, BP, DI y SI en la pila
PUSHAD	Salva el contenido de los registros EAX, ECX, D/X, EBX, ESP, EBP, EDI y ESI en la pila
POPA	Recupera datos de la pila y los carga en SI, DI, BP, SP, BX, DX, CX y AX
POPAD	Recupera datos de la pila y los carga en ESI , EDI, EBP, ESP, EBX, EDX, ECX y EAX
POP EAX	Recupera datos de la pila y los carga en EAX
PUSH EDI	Salva el contenido de EDI en la pila

Las instrucciones PUSHA y POPA salvan o recuperan todos los registros excepto los de segmento en la pila. Estas instrucciones no están disponibles en los primeros microprocesadores 8086/8088. La instrucción PUSH inmediato también es nueva en los microprocesadores 80286-80486. Consulte los ejemplos de la tabla 2-11 que muestran el orden de los registros transferidos por las instrucciones PUSHA y POPA. El 80386/80486 también permite salvar o recuperar los registros ampliados.

2-11 RESUMEN

1. Los modos de direccionamiento de datos incluyen: registro, inmediato, directo, indirecto por registro, base más índice, relativo a registro y direccionamiento relativo base más índice. En los microprocesadores 80386/80486 hay un modo adicional de direccionamiento llamado de índice escalado.
2. Los modos de direccionamiento a la memoria de programa incluyen: directo, relativo e indirecto
3. En la tabla 2-12 se enumeran todos los modos de direccionamiento de datos en modo real disponibles en los microprocesadores 8086-80486. En el modo protegido, la función del registro de segmento es direccionar a un descriptor que contiene la dirección base del segmento de memoria.
4. Los microprocesadores 80386 y 80486 tienen modos adicionales de direccionamiento que permiten que los registros ampliados EAX, EBX, ECX, EDX, EBP, EDI y ESI direccionen a la memoria. Estos modos de direccionamiento son muy numerosos como para incluirlos en una tabla, pero en general cualquiera de estos registros funciona en la misma forma que los de la tabla 2-12. Por ejemplo, MOV AL,TABL[EBX+2*ECX+10H] es un modo válido de direccionamiento para los microprocesadores 80386 y 80486.
5. La instrucción MOV copia el contenido del operando fuente en el operando de destino. La fuente nunca cambia con ninguna instrucción.
6. El direccionamiento por registros especifica cualquier registro de 8 bits (AH, AL, BH, BL, CH, CL, DH o DL) o cualquier registro de 16 bits (AX, BX, CX, DX, SP, BP, SI o DI). También se pueden direccionar los registros de segmentos (CS, DS, ES o SS) para transferir datos entre un registro de segmento y un registro o una localidad memoria de 16 bits o bien, mediante PUSH y POP. En los microprocesadores 80386 y 80486, los registros ampliados también se utilizan para direccionamiento de registros y son EAX, EBX, ECX, EDX, ESP, EBP, EDI y ESI; para estos microprocesadores están disponibles también los registros de segmento FS y GS.
7. La instrucción MOV que utiliza direccionamiento inmediato, transfiere el byte o palabra inmediatamente al código de operación hacia un registro o una localidad de memoria. El direccionamiento inmediato "manipula" datos constantes en un programa. En los 80386 y 80486, se pueden cargar datos inmediatos de doble palabra en un registro o localidad de memoria de 32 bits.
8. El direccionamiento directo ocurre en dos formas en el microprocesador: (1) direccionamiento directo y (2) direccionamiento por desplazamiento. Estas formas de direccionamiento son iguales, con la excepción de que el direccionamiento directo se emplea para transferir datos entre AX o AL y la memoria, mientras que el direccionamiento por desplazamiento se utiliza en cualquier transferencia de registro o memoria. El direccionamiento directo requiere 3 bytes de memoria; el de desplazamiento, requiere 4.

TABLA 2–12 Modos de direccionamiento en tiempo real de 8086-80486

Lenguaje ensamblador	*Generación de dirección*
MOV AL,BL	Direccionamiento de registro de 8 bits
MOB DI,BP	Direccionamiento de registro de 16 bits
MOV DS,BX	Segmento de direccionamiento de registro
MOV AL,LIST	(DS × 10H) + LIST
MOV CH,DATA1	(DS × 10H) + DATA1
MOV DS,DATA2	(DS × 10H) + DATA2
MOV AL,12	Immediate data of 12 decimal
MOV AL,[BP]	(SS × 10H) + BP
MOV AL,[BX]	(DS × 10H) + BX
MOV AL,[DI]	(DS × 10H) + DI
MOV AL,[SI]	(DS × 10H) + SI
MOV AL,[BP+2]	(SS × 10H) + BP + 2
MOV AL,[BX-4]	(DS × 10H) + BX - 4
MOV AL,[DI+1000H]	(DS × 10H) + DI + 1000H
MOV AL,[SI+300H]	(DS × 10H) + SI + 0300H
MOV AL,LIST[BP]	(SS × 10H) + BP + LIST
MOV AL,LIST[BX]	(DS × 10H) + BX + LIST
MOV AL,LIST[DI]	(DS × 10H) + DI + LIST
MOV AL,LIST[SI]	(DS × 10H) + SI + LIST
MOV AL,LIST[BP+2]	(SS × 10H) + BP + LIST + 2
MOV AL,LIST[BX-6]	(DS × 10H) + BX + LIST - 6
MOV AL,LIST[DI+100H]	(DS × 10H) + DI + LIST + 100H
MOV AL,LIST[SI+20H]	(DS × 10H) + SI + LIST + 20H
MOV AL,[BP+DI]	(SS × 10H) + BP + DI
MOV AL,[BP+SI]	(SS × 10H) + BP + SI
MOV AL,[BX+DI]	(DS × 10H) + BX + DI
MOV AL,[BX+SI]	(DS × 10H) + BX + SI
MOV AL,[BP+DI+2]	(SS × 10H) + BP + DI + 2
MOV AL,[BP+SI-4]	(SS × 10H) + BP + SI - 4
MOV AL,[BX+DI+30H)	(DS × 10H) + BX + DI + 30H
MOV AL,[BX+SI+10H]	(DS × 10H) + BX + SI + 10H
MOV AL,LIST[BP+DI]	(SS × 10H) + BP + DI + LIST
MOV AL,LIST[BP+SI]	(SS × 10H) + BP + SI + LIST
MOV AL,LIST[BX+DI]	(DS × 10H) + BX + DI + LIST
MOV AL,LIST[BX+SI]	(DS × 10H) + BX + SI + LIST
MOV AL,LIST[BP+DI+2]	(SS × 10H) + BP + DI + LIST + 2
MOV AL,LIST[BP+SI-7]	(SS × 10H) + BP + SI + LIST - 7
MOV AL,LIST[BX+DI-10H]	(DS × 10H) + BX + DI + LIST - 10H
MOV AL,LIST[BX+SI+1AFH]	(DS × 10H) + BX + SI + LIST + 1AFH

9. El direccionamiento indirecto por registro permite direccionar datos en la localidad de memoria a la que apunta un registro base (BP y BX) o un índice (DI y SI). En los 80386 y 80486 se utilizan registros ampliados EAX, EBX, ECX, EDX, EBP, EDI y ESI para direccionar datos en la memoria.

10. El direccionamiento base más índice, a menudo direcciona los datos en un arreglo. Para formar la dirección de la memoria para este modo, se suman un registro base, el registro índice y el contenido de un registro de segmento multiplicado por 10H. En los 80386 y 80486, los registros base e índice pueden ser cualquier registro de 32 bits, excepto EIP y ESP.
11. En el direccionamiento relativo por registro se utilizan una base o un registro índice, así como un desplazamiento, para accesar a los datos en memoria.
12. El direccionamiento relativo base más índice es útil para direccionar un arreglo bidimensional en la memoria. Para formar la dirección se suman un registro base, un registro índice, desplazamiento y el contenido de un registro de segmento multiplicado por 10H.
13. El direccionamiento de índice escalado es exclusivo de los microprocesadores 80386 y 80486. El segundo de dos registros (índice) se escala por un factor de 2X, 4X u 8X para accesar a palabras, dobles palabras o cuádruples palabras en los arreglos memoria.
14. El direccionamiento directo a la memoria de programa se puede efectuar con las instrucciones JMP y CALL a cualquier localidad en el sistema de memoria. Con este modo de direccionamiento, la dirección de desplazamiento y la dirección del segmento se almacenan con la instrucción.
15. El direccionamiento relativo al programa permite que una instrucción JMP o CALL transfiera el control del programa hacia adelante o atrás en el segmento de código en uso por ±32K bytes. En los 80386 y 80486, el desplazamiento de 32 bits permite la transferencia del programa a cualquier localidad del código de segmento actual con el empleo de un valor de desplazamiento de ±2G bytes.
16. El direccionamiento indirecto al programa permite que las instrucciones JMP o CALL direccionen otra parte del programa o subrutina en forma indirecta, a través de un registro o una localidad de memoria.
17. Las instrucciones PUSH y POP transfieren una palabra entre la pila y un registro o una localidad de la memoria. Está disponible una instrucción inmediata PUSH para colocar datos inmediatos en la pila. Las instrucciones PUSHA y POPA transfieren AX, CX, DX, BX, BP, SP, SI y DI entre la pila y estos registros. En los 80386 y 80486 el registro extendido de banderas y los extendidos también se pueden transferir entre registros y la pila. Una PUSHFD almacena al registro EFLAGS y la PUSHF almacena al registro FLAGS.

2–12 CUESTIONARIO Y PROBLEMAS

1. ¿Qué logran las siguientes instrucciones MOV?
   (a) MOV AX,BX
   (b) MOV BX,AX
   (c) MOV BL,CH
   (d) MOV ESP,EBP
   (e) MOV AX,CS
2. Enumere los registros de 8 bits que se utilizan para direccionamiento de registros.
3. Enumere los registros de 16 bits que se utilizan para direccionamiento de registros.
4. Enumere los registros de 32 bits que se utilizan para direccionamiento de los registros en los microprocesadores 80386 y 80486.

5. Enumere los registros de segmento de 16 bits utilizados con el direccionamiento de registros con MOV, PUSH y POP.
6. ¿Qué hay de incorrecto en la instrucción MOV BL,CX?
7. ¿Qué hay de incorrecto en la instrucción MV DS,SS?
8. Seleccione una instrucción para cada una de las siguientes tareas:
   (a) cargar EBX en EDX
   (b) cargar BL en CL
   (c) cargar SI en BX
   (d) cargar DS en AX
   (e) cargar AL en AH
9. Seleccione una instrucción para cada una de las siguientes tareas:
   (a) mover de 12H hacia AL
   (b) mover de 123AH hacia AX
   (c) mover de 0CDH hacia CL
   (d) mover de 1000H hacia SI
   (e) mover de 1200A2H hacia EBX
10. ¿Qué símbolo especial se utiliza, a veces, para denotar datos inmediatos?
11. ¿Qué es un desplazamiento? ¿Cómo determina la dirección de la memoria en una instrucción MOV [2000H],AL?
12. ¿Qué indican los símbolos (corchetes) []?
13. ¿Qué indica la dirección 2000:3000?
14. Supóngase que DS = 0200H, BX = 0300H y DI= 400H. Determine la dirección de la memoria a la cual se accesa con cada una de lassiguientes instrucciones, en el supuesto de funcionamiento en modo real:
   (a) MOV AL,[1234H]
   (b) MOV EAX,[BX]
   (c) MOV [DI],AL
15. ¿Qué hay de incorrecto en una instrucción MOV [BX],[DI]?
16. Seleccione una instrucción que requiera BYTE PTR.
17. Seleccione una instrucción que requiera WORD PTR.
18. Seleccione una instrucción que requiera DWORD PTR.
19. Explique la diferencia entre la instrucción MOV BX,DATO y la instrucción MOV BX,OFFSET DATO.
20. Dado que DS = 1000H, SS = 2000H, BP = 1000H y DI = 0100H, determine la dirección de la memoria que accesa cada una de las siguientes instrucciones en el supuesto de funcionamiento en modo real.
   (a) MOV AL,[BP + DI]
   (b) MOV CX,[DI]
   (c) MOV EDCX,[BP]
21. Dado que DS = 1200H, BX = 0100H y SI = 0250H, determine la dirección de la memoria a que accede cada una de las siguientes instrucciones, en el supuesto de funcionamiento en modo real:
   (a) MOV [100H],DL
   (b) MOV [SI +100H)],EAX
   (c) MOV DL,[BX + 100H]

22. Dado que DS = 1100H, BX = 0200H, LISTA = 0250H y SI = 0500H, determine la dirección de la memoria que accesa cada una de lassiguientes instrucciones, en el supuesto de funcionamiento en modo real:
    (a) MOV LISTA[SI],EDX
    (b) MOV CL,LISTA [BX +SI
    (c) MOV CH,[BX +SI]
23. Dado que DS = 1300H, SS = 1400H, BP = 1500H y SI = 0100H, determine la dirección de la memoria a que accesa cada una de las siguientes instrucciones, en el supuesto de funcionamiento en modo real:
    (a) MOV EAX,[BP + 200H]
    (b) MOV AL,[BP + SI - 200H]
    (c) MOV AL,[SI-0100H]
24. ¿Cuál registro de base direcciona los datos en el segmento de pila?
25. Dado que EAX = 00001000H, EBX = 00002000H y DS = 0010H, determine las direcciones que accesan cada una de las siguientes instrucciones en el supuesto de funcionamiento en modo real:
    (a) MOV ECX,[EAX + EBX]
    (b) MOV [EAX +2*EBX],CL
    (c) MOV DH,[EBX +4*EAX +1000h]
26. Enumere los tres modos de direccionamiento de la memoria de programa.
27. ¿Cuántos bytes de memoria almacenan una instrucción para salto directo lejano? ¿Qué se almacena en cada uno de estos bytes?
28. ¿Cuál es la diferencia entre un salto entresegmentos y uno intrasegmento?
29. Si en el salto cercano se utiliza un desplazamiento de 16 bits con signo, ¿cómo puede brincar a cualquier localidad de memoria dentro del segmento de código actual?
30. ¿Qué es un brinco largo o lejano?
31. Si se almacena una instrucción JMP en la localidad 100H de la memoria dentro del segmento de código actual, no puede ser un salto __________ si va a saltar a una localidad 200H en la memoria, dentro del segmento actual de código.
32. Muestre cuál instrucción JMP ensambla (corta, cercana o lejana) si la instrucción JMP ALLA está almacenada en una dirección 10000H en la memoria y si la dirección de ALLA es:
    (a) 10020H
    (b) 11000H
    (c) 0FFFEH
    (d) 30000H
33. Forme una instrucción JMP que brinque a la dirección a que apunta el registro BX.
34. Seleccione una instrucción JMP que brinque a una localidad ación almacenada en la memoria en una tabla de localidad. Supóngase que es un JMP cercano.
35. ¿Cuántos bytes se salvan en la pila con las instrucciones PUSH?
36. Explique cómo funciona la instrucción PUSH[DI].
37. ¿Qué registros y en qué orden los coloca en la pila la instrucción PUSHA?
38. ¿Qué logra la instrucción PUSHAD?

CAPITULO 3

Instrucciones para transferencia de datos

INTRODUCCION

En este capítulo se explican las instrucciones para transferencia de datos en los 8086-80486. Las instrucciones para transferencia de datos incluyen MOV, MOVSX, MOVZX, PUSH, POP, BSWAP, XCHG, XLAT, IN, OUT, LEA, LDS, LES, LFS, LGS, LSS, LAHF, SAHF y las instrucciones para cadena MOVS, LODS, STOS, INS y OUTS. Se presentan primero las instrucciones para la transferencia de datos, porque son las que se emplean más a menudo y son más fáciles de entender.

El microprocesador requiere un programa ensamblador, que genera lenguaje de máquina, porque las instrucciones en lenguaje de máquina son muy complejas para generarlas a mano. En este capítulo se describen la sintaxis y algunos de los directivos del lenguaje ensamblador. (Para el texto de esta sección se supone que el usuario está desarrollando la programación en una computadora personal IBM o compatible, con el empleo del MACRO ensamblador de Microsoft, el Ensamblador de Intel (ASM) el turboensamblador de Borland (TASM) o software similar. Se presenta información que funciona con el ensamblador MASM de Microsoft, pero casi todos los programas funcionan sin cambio con el empleo de otros ensambladores. En el apéndice A se explica el ensamblador Microsoft y se dan detalles del programa ligador.)

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo el lector podrá:

1. Explicar el funcionamiento de cada instrucción para transferencia de datos con los modos de direccionamiento correspondientes.
2. Explicar la finalidad de las siguientes pseudooperaciones y palabras clave del lenguaje ensamblador, tales como ALIGN, ASSUME, DB, DD, DW, END, ENDS, ENDP, EQU, OFFSET, ORG, PROC, PTR, USE16, USE32 y SEGMENT.
3. Dada una tarea específica para transferencia de datos, seleccionar las instrucciones adecuadas en lenguaje ensamblador para efectuarla.

4. Dada una instrucción de lenguaje de máquina hexadecimal determinar el código de operación simbólico, la fuente, el destino y el modo de direccionamiento.
5. Utilizar el ensamblador para inicializar un segmento de datos, un segmento de pila y un segmento de código.
6. Mostrar cómo iniciar un procedimiento con PROC y con ENDP.
7. Explicar la diferencia entre las definiciones de modelos de memoria y segmento completo para el ensamblador MASM.

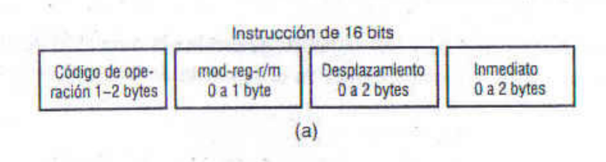
3–1 DE NUEVO CON MOV

En el capítulo 2 se utilizó la instrucción MOV para explicar los diversos modos de direccionamiento para los 8086-80486. En este capítulo, se utiliza MOV para presentar las instrucciones en lenguaje de máquina disponibles para el programador, para varios modos de direccionamiento e instrucciones. Se da a conocer el código de máquina porque, a veces, puede ser necesario para interpretar programas en lenguaje de máquina producidos por un ensamblador. La interpretación del lenguaje de máquina permite la depuración o modificación en el entorno del lenguaje de máquina. También se presenta la forma de convertir entre instrucciones en lenguaje de máquina y ensamblador con el empleo del apéndice B.

Lenguaje de máquina

El **lenguaje de máquina** es el código binario nativo que entiende el microprocesador y lo utiliza en las instrucciones que controlan su funcionamiento. La longitud de las instrucciones en lenguaje de máquina para los 8086-80486 varía entre 1 y 13 bytes. Aunque el lenguaje de máquina parece ser complejo, hay orden en este lenguaje para microprocesador. Hay más de 20,000 variantes en el lenguaje de máquina para los microprocesadores 8086-80486, lo cual significa que no hay una lista completa de estas variantes. Debido a ello, sólo se dan algunos bits en una instrucción en lenguaje de máquina y el resto se debe determinar para cada variante de la instrucción.

Las instrucciones para los 8086-80286 son en el modo de 16 bits que tienen la configuración que se ilustra en la figura 3-1(a). Estas instrucciones en el modo de 16 bits son compatibles con los microprocesadores 80386 y 80486, si están inicializados para utilizar formatos de instruccio-



Instrucción de 32 bits (sólo 80386 u 80486)

Tamaño de dirección 0 a 1 byte	Tamaño de operando 0 a 1 byte	Código de operación 1 a 2 bytes	mod-reg-r/m 0 a 1 byte	Indice escalado 0 a 1 byte	Desplazamiento 0 a 4 bytes	Inmediato 0 a 4 bytes

(b)

FIGURA 3–1 Los formatos de las instrucciones para 8086-80486. (a) La forma de 16 bits; (b) la forma de 32 bits.

nes en el modo de 16 bits. En los 80386 y 80486 se supone que todas las instrucciones lo son en el modo de 16 bits cuando la máquina trabaja en modo real. En el modo protegido, el byte superior del descriptor del 80386 y 80486 contiene el bit D que selecciona el modo de instrucción de 16 o de 32 bits. En esos microprocesadores se utilizan también instrucciones en modo de 32 bits con la configuración que se ilustra en la figura 3-1(b). Estas instrucciones pueden estar en el modo de instrucción de 16 bits, mediante el empleo de prefijos.

Los primeros 2 bytes del formato del modo de instrucción de 32 bits, se llaman prefijos de cambio, porque no siempre están presentes. El primer byte modifica el tamaño de la dirección utilizada por la instrucción y el segundo modifica el tamaño del registro. Si los 80386 y 80486 funcionan con instrucciones de máquina del modo de 16 bits (en modo real o protegido) y se utiliza un registro de 32 bits, se coloca el prefijo (66H) del tamaño del operando al principio de la instrucción. Si los 80386 y 80486 funcionan con instrucciones de máquina del modo de instrucción de 32 bits (sólo en modo protegido) y se utiliza un registro de 32 bits, no está presente el prefijo del tamaño del operando. Si aparece un registro de 16 bits en la instrucción, en el modo de instrucción de 32 bits, está presente el prefijo de tamaño del registro para seleccionar un registro de 16 bits. El prefijo del tamaño de dirección se utiliza en forma similar, como se explicará más adelante. El prefijo cambia el tamaño del registro y del operando de 16 bits a 32 bits o viceversa, para la instrucción con prefijo.

El código de instrucción. Selecciona la operación (suma, resta, transferencia, etc.) que ejecuta el microprocesador. El código de instrucción tiene uno o dos bytes de longitud para las instrucciones en lenguaje de máquina. En la figura 3-2 se ilustra la configuración general del primer byte del código de operación de muchas, pero no todas, las instrucciones en lenguaje de máquina. En este caso, los 6 primeros bits del primer byte, son el código en binario. Los 2 bits restantes indican el *sentido* (D) del flujo de datos y éstos son un *byte* o una *palabra* (W). En los 80386 y 80486 se especifican palabras y dobles palabras cuando W = 1. El modo de instrucción y el prefijo del tamaño del operando seleccionan si W representa una palabra o una doble palabra en los microprocesadores 80386 y 80486.

Si el bit D = 1 los datos fluyen al campo del registro (REG) desde el campo R/M en el siguiente byte de la instrucción. Si el bit D = 0, los datos fluyen al campo REG del campo R/M. Si el bit W = 1, el tamaño de los datos es una palabra o doble palabra; si el bit W = 0, el tamaño de los datos es un byte. El bit W aparece en la mayor parte de las instrucciones, mientras que el bit D casi siempre aparece con MOV y unas cuantas instrucciones más. En la figura 3-3 aparece el patrón de bits binarios del segundo byte del código de operación (reg-mod-r/m) de muchas instrucciones. En esta figura se muestra la ubicación de los campos REG (*registro*), R/M (*registro/memoria*) y MOD (*modo*).

Campo MOD. El *campo MOD* selecciona el modo de direccionamiento y si hay desplazamiento en el modo seleccionado. En la tabla 3-1 se presentan las formas de operandos disponibles para el campo MOD, para el modo de instrucción de 16 bits en los microprocesadores 8086-80486. Si



FIGURA 3-2 El byte 1 de muchas instrucciones en lenguaje de máquina ilustrando la posición del código de operación, D y W.

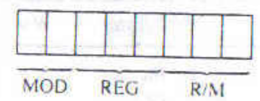


FIGURA 3–3 El byte 2 de muchas instrucciones en lenguaje de máquina ilustrando la posición de los campos MOD, REG Y R/M.

TABLA 3–1 Especificaciones del campo MOD para el modo de instrucción de 16 bits

MOD	*Función*
00	No hay desplazamiento
01	Desplazamiento extendido por signo de 8 bits
10	Desplazamiento de 16 bits
11	R/M es un registro

el campo MOD contiene un 11, selecciona el modo de direccionamiento de registro, en el cual se utiliza el campo R/M para especificar un registro en lugar de una localidad de la memoria. Si el campo MOD contiene un 00, 01 o 10, el campo R/M selecciona uno de los modos de direccionamiento para la memoria de datos. Cuando MOD selecciona un modo de direccionamiento para la memoria, indica que el modo de direccionamiento no contiene desplazamiento (00), un desplazamiento de 8 bits con signo extendido (01) o un desplazamiento de 16 bits (10). La instrucción MOV AL, [DI], es un ejemplo en el cual no se utiliza desplazamiento. En una instrucción MOV AL, [DI+2], se utiliza un desplazamiento de 8 bits (+2) y en una instrucción MOV AL, [DI + 1000] se utiliza un desplazamiento de 16 bits (+1000).

Todos los desplazamientos de 8 bits con signo a 16 bits cuando el microprocesador ejecuta la instrucción. Si el desplazamiento de 8 bits es 00H-7FH (positivo), se extiende por signo a 0000H-007FH antes de sumarlo a una dirección de segmento. Si el desplazamiento de 8 bits es 80H-FFH (negativo) se extiende por signo a FF80H-FFFFH. Para extender por signo un número, se copia su bit de signo en el siguiente byte de orden superior, con lo cual se genera un 00H o un FFH en el byte de orden superior.

En los microprocesadores 80386/80486 el campo MOD puede ser el mismo que el de la tabla 3-1 o, si el modo de instrucción es de 32 bits, es el de la tabla 3-2. El campo MOD se interpreta como seleccionado por el prefijo de cambio de dirección o por el modo de funcionamiento del microprocesador. Este cambio en la interpretación del campo MOD y en la instrucción soporta numerosos modos de direccionamiento adicionales permitidos en los microprocesadores 80386 y 80486. La diferencia principal es cuando el campo MOD es un 10. Esto hace que el desplazamiento de 16 bits se convierta en uno de 32 bits para permitir accesar a cualquier locali-

TABLA 3–2 Especificaciones del campo MOD para el modo de instrucción de 32 bits (sólo 80386/80486)

MOD	*Función*
00	No hay desplazamiento
01	Desplazamiento extendido por signo de 8 bits
10	Desplazamiento de 32 bits
11	R/M es un registro

TABLA 3–3 Asignaciones a REG y R/M (cuando MOD = 11)

Código	*W = 0 (byte)*	*W = 1 (palabra)*	*W = 1 (doble palabra)*
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

dad de memoria (4G bytes) en el modo protegido. Los 80386 y 80486 sólo permiten un desplazamiento de 8 o de 32 bits cuando funcionan en el modo de instrucción de 32 bits.

Asignaciones de registros. En la tabla 3-3 aparecen las asignaciones de registros para el campo REG y el campo R/M (MOD = 11). La tabla contiene tres listas de asignaciones de registros: una se emplea cuando el bit W = 0 y las otras dos cuando el bit W = 1 (palabras o dobles palabras). Se debe tener en cuenta que los registros de doble palabra sólo están disponibles en los microprocesadores 80386 y 80486.

Suponga que aparece una instrucción de 2 bytes 8BECH en un programa en lenguaje de máquina. Debido a que no aparecen ni 66H (prefijo de cambio de tamaño del operando) ni 67H (prefijo de cambio del tamaño del registro) como primer byte, entonces el primer byte es el código de operación. Si se supone que el microprocesador funciona en el modo de instrucción de 16 bits, esta instrucción se convierte en binario y se coloca en el formato de instrucción de los bytes 1 y 2 como se ilustra en la figura 3-4. El código de operación es 100010. En el apéndice B en donde se enumeran las instrucciones en lenguaje de máquina, se verá que se trata del código de una instrucción MOV. Además, se verá que los bits D y W son uno, lo cual significa que se transfiere una palabra al registro especificado en el campo REG; éste contiene 101 indicando el registro BP, por lo cual la instrucción MOV transfiere datos al registro BP. Debido a que el campo MOD contiene un 11, el campo R/M también indica un registro. En este caso, R/M = 100 (SP); por lo tanto, la instrucción transfiere los datos SP a BP lo que se escribe en forma simbólica como la instrucción MOV BP,SP.

Supóngase que aparece una instrucción 668BE8H en un 80386 u 80486 que funcione en el modo de instrucción de 16 bits. El primer byte (66) es el prefijo de cambio del tamaño del operan-

FIGURA 3–4 La instrucción 8BEC colocada en los formatos de bytes 1 y 2 de las figuras 3-1 y 3-2. Esta instrucción en lenguaje de máquina es el código de la instrucción simbólica MOV BP,SP.

Código de operación						D	W	MOD		REG			R/M		
1	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0

Código de operación = MOV
D = Registro (REG)
W = Palabra
MOD = R/M es registro
REP = BP
R/M = SP

TABLA 3–4 Modos de direccionamiento de 16 bits de memoria R/M

Código	*Función*
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[SI]
110	SS:[BP*]
111	DS:[BX]

******Nota*: Véase Modos Especiales de Direccionamiento en el texto.

do que indica que el código es el de MOV con EAX como operando fuente y EBP como operando destino. Esta instrucción es MOV EBP,EAX la cual se convierte en MOV BP,AX en los microprocesadores 80386/80486, si trabajan en el modo de instrucción de 32 bits. Por fortuna, el programa del ensamblador mantiene el control de los prefijos de tamaño de operando y de dirección.

Direccionamiento de la memoria R/M. Si el campo MOD contiene un 00, 01 o 10, el campo R/M adquiere un nuevo significado. En la tabla 3-4 aparecen los modos de direccionamiento de la memoria que da el campo R/M cuando MOD es 00, 01 o 10 para el modo de instrucción de 16 bits.

Todos los modos de direccionamiento de 16 bits descritos en el capítulo 2 aparecen en la tabla 3-4. El desplazamiento, que se comentó en el capítulo 2, se define con el campo MOD. Si MOD = 00 y R/M = 101, el modo de direccionamiento es [DI]. Si MOD = 01 o 10, el modo de direccionamiento es [DI + 33H] o LIST [DI + 22H] para el modo de instrucción de 16 bits. En este ejemplo se emplean LIST, 33H y 22H como valores arbitrarios para el desplazamiento.

En la figura 3-5 se ilustra la versión en lenguaje de máquina de la instrucción de 16 bits MOV DL,[DI] o la instrucción (8A15H). Esta instrucción tiene 2 bytes de longitud y tiene un código 100010, D = 1 (*a REG desde R/M*), W = 0 (byte), MOD = 00 (*no hay desplazamiento*) REG = 010 (*DL*) R/M = 101 (*[DI]*). Si la instrucción cambia a MOV DL, [DI + 1], el campo MOD cambia a 01 para un desplazamiento de 8 bits, pero, por lo demás, los 2 primeros bytes de la instrucción siguen iguales. La instrucción se vuelve ahora 8A5501H en lugar de 8A15H. Se debe

FIGURA 3–5 Una instrucción MOV DL,[DI] codificada en lenguaje binario de máquina.

Código de operación						D	W	MOD		REG			R/M		
1	0	0	0	1	0	1	0	0	0	0	1	0	1	0	1

Código de operación = MOV
D = Registro (REG)
W = Byte
MOD = No hay desplazamiento
REG = DI
RIM = DI

tener en cuenta que el desplazamiento de 8 bits se añade a los primeros 2 bytes de la instrucción para formar una instrucción de 3 bytes en vez de una de 2 bytes. Si se vuelve a cambiar la instrucción a una MOV DL,[DI+1000H] la forma en lenguaje de máquina se vuelve 8A750010H. En este caso, el desplazamiento de 16 bits de 1000H (codificado como 0010H), se añade al código de operación.

Modo especial de direccionamiento. Hay un modo especial de direccionamiento (para las instrucciones de 16 bits) que no aparece en las tablas 3-3, 3-4 o 3-5 y que ocurre siempre que se consultan los datos de la memoria con sólo el modo de direccionamiento por desplazamiento en las instrucciones de 16 bits. Los ejemplos son MOV [1000H], DL y MOV NUMB,DL. La primera instrucción transfiere el contenido del registro DL hacia la localidad 1000H de la memoria en el segmento de datos. La segunda instrucción transfiere el contenido del registro DL a la localidad NUMB del segmento de datos.

Siempre que la instrucción tiene un solo desplazamiento, el campo MOD es 00 y el campo R/M es siempre 110. Esta combinación por lo general muestra que la instrucción no contiene desplazamiento y se utiliza el modo de direccionamiento indirecto [BP]. En la práctica no se puede emplear el modo de direccionamiento [BP] sin un desplazamiento en lenguaje de máquina. El ensamblador se encarga de ello porque utiliza un desplazamiento de 8 bits (MOD = 01) de 00H siempre que el modo de direccionamiento [BP] aparece en una instrucción. Esto significa que el modo de direccionamiento [BP] se ensambla como [BP + 0] aunque se utilice [BP].

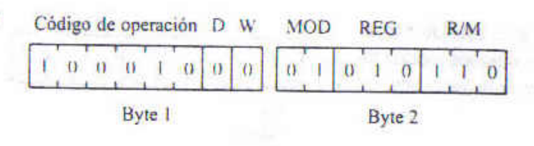
En la figura 3-6 se muestra el patrón de bits binarios que se requieren para codificar la instrucción MOV[1000H],DL en lenguaje de máquina. Si la persona que traduce esta instrucción simbólica al lenguaje de máquina no sabe el modo especial de direccionamiento, la traduciría en forma incorrecta como una instrucción MOV [BP],DL. En la figura 3-7 se ilustra la forma real de la instrucción MOV [BP],DL; note que es una instrucción de 3 bytes con un desplazamiento de 00H.

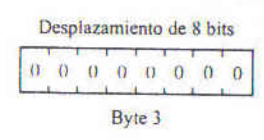
Modos de direccionamiento de 32 bits. Para obtener los modos de direccionamiento de 32 bits que hay en los microprocesadores 80386/80486, se hace trabajar estas máquinas en el modo de instrucción de 32 bits o en el modo de instrucción de 16 bits, con el empleo del prefijo (67H) de tamaño de dirección. En la tabla 3-5 se muestra la codificación de R/M utilizada para especificar los modos de direccionamiento de 32 bits. se verá que cuando R/M = 100, aparece en la instrucción un byte adicional llamado *byte de índice escalado*, el cual indica las formas adicionales de direccionamiento de índice escalado que no aparecen en la tabla 3-5. El empleo principal del byte de índice escalado es cuando se suman dos registros para especificar la dirección de la memoria en una instrucción.

FIGURA 3–6 Una instrucción MOV[1000H],DL codificada en binario. Se verá que se requieren 2 bytes adicionales para el desplazamiento de 1000H.



FIGURA 3–7 La instrucción MOV [BP],DL codificada en binario requiere un byte de desplazamiento de 00H a fin de que los 8086/8088 ejecuten esta instrucción.





En la figura 3-8 se muestra el formato del *byte de índice escalado* seleccionado por un valor de 100 en el campo R/M de una instrucción, cuando se utiliza una dirección de 32 bits en los 80386/80486. Los dos bits que están más a la izquierda seleccionan un factor de escala (multiplicador) de 1X, 2X u 8X. Se debe tener en cuenta que un factor de escala de 1X está implícito si no se utiliza ninguno en una instrucción que contiene dos registros de direccionamiento indirecto de 32 bits. Ambos campos de índice y de base contienen los números de registro indicados en la tabla 3-3 para los registros de 32 bits.

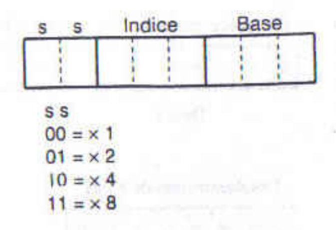
La instrucción MOV EAX [EBX + 4*ECX] se codifica como 67668B048BH. Se debe tener en cuenta que ambos prefijos de cambio de tamaño de dirección (67H) y del tamaño del registro (66H) aparecen en esta instrucción. Esto significa que la instrucción es 67668B048BH cuando se trabajan los 80386/80486 en el modo de instrucción de 16 bits; si funcionan en el modo de instrucción de 32 bits, desaparecen ambos prefijos, por lo cual la instrucción se convierte en 8B048BH. El empleo de los prefijos depende del modo de funcionamiento de los microprocesadores 80386 y 80486.

TABLA 3–5 Modos de direccionamiento de 32 bits seleccionados por R/M

Código	*Función*
000	DS:[EAX]
001	DS:[ECX]
010	DS:[EDX]
011	DS:[EBX]
100	Se emplea bit de índice escalado
101	SS:[EBP*]
110	DS:[ESI]
111	DS:[EDI]

**Nota:* Si los bits de MOD son 00, en este modo de direccionamiento se emplea un desplazamiento de 32 bits sin el registro EBP. Es semejante al modo especial de direccionamiento de 16 bits.

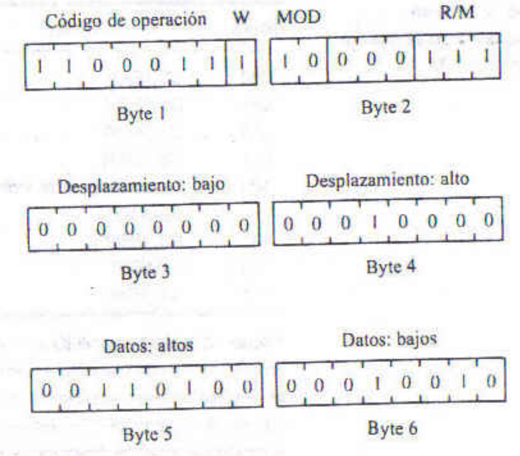
FIGURA 3–8 El byte de índice escalado de algunos modos de direccionamiento de 32 bits.



Una instrucción inmediata. Suponga que se escoge la instrucción MOV WORD PTR [BX+ 1000H],1234H como ejemplo de una instrucción de 16 bits que emplea el direccionamiento inmediato. Esta instrucción transfiere 1234H a la localidad de memoria de tamaño de palabra direccionada por la suma de 1000H, BX y DS × 10H. En esta instrucción de 6 bytes se emplean 2 bytes para el código de operación, D, W, y los campos MOD, REG y R/M. Dos de los 6 bytes son los datos 1234H. Otros dos de los 6 bytes son el desplazamiento de 1000. En la figura 3-9 se muestran los patrones de bits para cada byte de esta instrucción.

En su forma simbólica esta instrucción incluye el directivo WORD PTR, la cual indica al ensamblador que en la instrucción se utiliza un apuntador a memoria de tamaño de palabra. Si la instrucción mueve un byte de datos inmediatos, entonces BYTE PTR sustituye a WORD PTR en la instrucción. Asimismo, si en la instrucción se emplea una doble palabra de datos inmediatos, el directivo DWORD PTR sustituye a BYTE PTR. Casi todas las instrucciones que hacen referencia a la memoria por medio de un apuntador no necesitan los directivos BYTE PTR, WORD PTR o DWORD PTR, pues sólo se necesitan cuando no está claro si la operación es con un byte o una palabra. La instrucción MOV [BX], AL por supuesto, es para transferir un byte; la instrucción MOV[BX], 1 no es exacta y podría ser una transferencia de un byte, de palabra o de doble palabra. En este caso, la instrucción se debe codificar como MOV BYTE PTR[BX],1, MOV WORD

FIGURA 3–9 Una instrucción MOV WORD PTR [BX + 1000H], 1234H convertida a binario, requiere 6 bytes de memoria: 2 para la instrucción, 2 para el desplazamiento de 1000H y 2 para los datos inmediatos de 1234H.



Nota: D y REG no se utilizan en el direccionamiento de memoria inmediato.

PTR [BX],1 o MOV DWORD PTR[BX],1. Si no, el ensamblador muestra error porque no puede determinar lo que se pretende con esa instrucción.

Instrucciones MOV para segmento. Si el contenido de un registro de segmento se transfiere con la instrucción MOV, PUSH o POP, entonces un grupo especial de bits de registro (campo REG) selecciona el registro de segmento (véase tabla 3-6).

En la figura 3-10 se muestra una instrucción MOV BX,CS convertida a binario. El código de operación para este tipo de instrucción MOV es diferente para las instrucciones MOV ya mencionadas. Los registros de segmento se pueden transferir entre cualquier registro de 16 bits o localidad de memoria de 16 bits. Por ejemplo, la instrucción MOV [DI],DS transfiere el contenido de DS a la localidad de memoria direccionada por DI en el segmento de datos.

Aunque en lo precedente no se ha hecho una cobertura completa de la codificación del lenguaje de máquina, sirve como un buen comienzo para programar en lenguaje de máquina. Recuerde que un programa escrito en lenguaje simbólico (lenguaje de ensamblador) rara vez se ensambla a mano a lenguaje de máquina binario. Un *ensamblador* convierte el lenguaje simbólico de ensamble en lenguaje de máquina. Con el microprocesador y sus más de 20,000 variantes en las instrucciones, sólo queda esperar que se cuente con un ensamblador para la conversión, porque este procedimiento consume mucho tiempo, pero no es imposible.

3–2 PUSH/POP

Las instrucciones PUSH y POP son importantes pues almacenan y recuperan datos de la memoria de pila LIFO por sus siglas en inglés (último en entrar, primero en salir). Los microprocesadores 8086-80486 tienen seis formas de instrucciones PUSH y POP: registro, memoria, inmediata, registro de segmento, banderas y todos los registros. Las formas PUSH y POP inmediatas y las formas PUSHA y POPA (todos los registros) no están disponibles en los primeros microprocesadores 8086-8088 pero sí en los 80286, 80386 y 80486.

El direccionamiento por registros permite transferir cualquier registro de 16 bits desde o hacia la pila. En los 80386/80486 los registros extendidos de 32 bits y las banderas (EFLAGS)

TABLA 3–6 Bits de selección de registro de segmento

Código	*Registro de segmento*
000	ES
001	CS*
010	SS
011	DS
100	FS
101	GS

**Nota*: El microprocesador no permite MOV CS, ?? ni POP CS. Los segmentos FS y GS sólo están disponibles para los microprocesadores 80386/80486.

Código de operación	MOD	REG	R/M
1 0 0 0 1 1 0 0	1 1	0 0 1	0 1 1

Nota: W y D no están presentes en esta instrucción.

FIGURA 3–10 Una instrucción MOV BX,CS convertida a binario. Aquí REG está codificado en el registro CS y R/M está codificado en el registro BX.

también pueden transferirse a y desde la pila. El direccionamiento a memoria almacena los contenidos de una localidad de memoria de 16 bits (32 bits en los 80386/80486) en la pila o transferir datos de la pila a una localidad de memoria. El direccionamiento inmediato permite salvar datos inmediatos dentro de la pila, pero no recuperarlos. El direccionamiento por registro de segmento permite salvar o recuperar la pila (CS se puede salvar pero los datos de la pila no se pueden recuperar hacia CS). Las banderas se pueden salvar en la pila o recuperar de ella así como el contenido de todos los registros.

Push

La instrucción PUSH para los 8086-80286, siempre transfiere 2 *bytes* de datos a la pila y en los 80386 y 80486 transfiere 2 o 4 bytes según sean el registro o el tamaño de la localidad de memoria. La fuente de los datos puede ser cualquier registro interno de 16 o de 32 bits, datos inmediatos, cualquier registro de segmento o 2 bytes cualesquiera de datos de la memoria. También hay una instrucción PUSHA que salva el contenido del grupo de registros internos, excepto el de segmento de pila, en la pila. La instrucción PUSHA (*salvar todo*) salva los registros en la pila en el siguiente orden: AX, CX, DX, BX, SP, BP, SI y DI. El valor de SP salvado en la pila el que había antes de ejecutar la instrucción. La instrucción PUSHF (*salvar banderas*) transfiere el contenido del registro de banderas a la pila. Las instrucciones PUSHAD y POPAD salvan y recuperan el contenido del grupo de registros de 32 bits que hay en los microprocesadores 80386 y 80486.

Siempre que se transfieren los datos a la pila, el primer byte de datos (el más significativo) se transfiere dentro de la localidad de la memoria del segmento de pila direccionada por SP – 1. El segundo byte de datos (el menos significativo) se transfiere a la localidad en la memoria de segmento de pila direccionada por SP – 2. Una vez que se transfieren los datos con PUSH, el contenido del registro SP decrementa en 2. Lo mismo ocurre al transferir una doble palabra, excepto que se transfieren 4 bytes a la memoria de pila (primero el byte más significativo), entonces el apuntador de la pila se decrementa en 4. En la figura 3-11 se muestra el funcionamiento de la instrucción PUSH AX, la cual transfiere el contenido de AX en la pila en la forma: SS:[SP – 1] = AH, SS:[SP – 2]) = AL y, después, SP = SP – 2.

La instrucción PUSHA transfiere todos los registros internos de 16 bits dentro de la pila, como se ilustra en la figura 3-12. Esta instrucción requiere 16 bytes de espacio en la memoria de pila para almacenar los ocho registros de 16 bits. Una vez transferidos todos los registros, el contenido del registro SP se decrementa en 16. La instrucción PUSHA es muy útil cuando hay que salvar la totalidad del grupo de registros (*entorno del microprocesador*) de los 80286-80486.

La instrucción PUSH para datos inmediatos tiene dos códigos de operación diferentes, pero en ambos casos el número inmediato de 16 bits se transfiere hacia dentro de la pila o, si se utiliza PUSHD, se transfiere un dato inmediato de 32 bits. Si el valor de los datos inmediatos es 00H-FFH, el código es 6AH; si los datos son 0100H-FFFFH, es 68H. La instrucción PUSH 8 se ensambla como 6A08H y la instrucción PUSH 1000H se ensambla como 680010H.

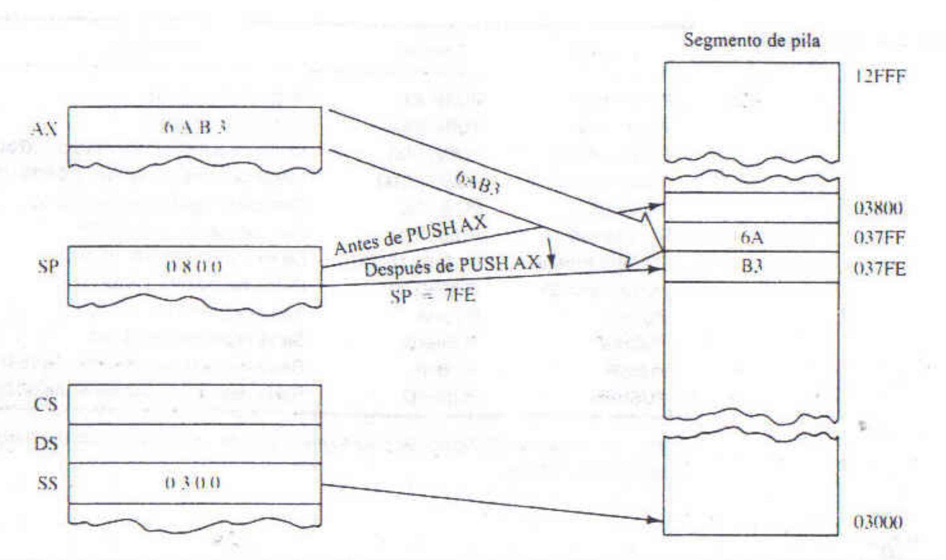


FIGURA 3–11 El efecto de una instrucción PUSH AX en el registro SP y en las localidades 037FFH y 037FEH de la memoria de pila.

En la tabla 3-7 se presentan las formas de la instrucción PUSH que incluyen PUSHA y PUSHF. Se verá la forma en que se emplea el conjunto de instrucciones para especificar diferentes tamaños de datos con el ensamblador.

FIGURA 3–12 La instrucción PUSHA transfiere a todos los registros internos de 16 bits hacia la pila.

16 bits
AX
CX
DX
BX
SP
BP
SI
DI
SP después de PUSHA →

TABLA 3–7 Las instrucciones PUSH

Simbólica	*Ejemplo*	*Funciones*
PUSH reg16	PUSH BX	Registro de 16 bits
PUSH reg32	PUSH EAX	Registro de 32 bits
PUSH mem16	PUSH [BX]	Modo de direccionamiento de 16 bits
PUSH mem32	PUSH [EAX]	Modo de direccionamiento de 32 bits
PUSH seg	PUSH DS	Cualquier registro de segmento
PUSH imm8	PUSH 12H	Datos inmediatos de 8 bits
PUSHW imm16	PUSHW 1000H	Datos inmediatos de 16 bits
PUSHD imm32	PUSHD 20	Datos inmediatos de 32 bits
PUSHA	PUSHA	Salva registros de 16 bits
PUSHAD	PUSHAD	Salva registros de 32 bits
PUSHF	PUSHF	Salva registro de banderas de 16 bits
PUSHFD	PUSHFD	Salva registro de banderas de 32 bits

Nota: Se requieren 80386/80486 para trabajar con direcciones, registros y datos inmediatos de 32 bits.

POP

La instrucción POP efectúa, a la inversa, las acciones de la instrucción PUSH. La instrucción POP transfiere datos de la pila y los carga en los destinos que pueden ser un registro de 16 bits, un registro de segmento o una localidad de 16 bits en la memoria. En los 80386/80486, POP también puede transferir datos de 32 bits de la pila y utiliza direcciones de 32 bits. La instrucción POP no está disponible como POP inmediato. La instrucción POPF (*recuperar las banderas*) transfiere un número de 16 bits de la pila a registro y la POPFD elimina un número de 32 bits de la pila y la coloca en el registro extendido de banderas. La instrucción POPA (*recuperar todos*) recupera 16 bytes de datos de la pila y los coloca en los registros siguientes en este orden: DI, SI, BP, SP, BX, DX, CX y AX, o sea en el orden inverso al cual los transfirió la instrucción PUSHA a la pila. En los 80386/80486, una instrucción POPAD recupera los registros de 32 bits desde la pila.

Supóngase que se ejecuta una instrucción POP BX. El primer byte de datos recuperados de la pila (la localidad de memoria direccionada por SP en el segmento de pila) se transfiere hacia el registro BL. El segundo byte se transfiere de la localidad SP+1 de la memoria del segmento de pila se coloca a registro BH. Después de haber transferido ambos bytes de la pila, el registro SP incrementa en 2. En la figura 3-13 se ilustra la forma en que la instrucción POP BX recupera los datos de la pila y los carga en el registro BX.

Los códigos de operación para la instrucción POP y todas sus variantes aparecen en la tabla 3-8. Se debe tener en cuenta que la instrucción POP CS no es válida en el conjunto de instrucciones para el 80286. Si se permite que se ejecute una instrucción POP CS sólo cambia una parte (CS) de la dirección de la siguiente instrucción. Esto hace que la instrucción POP CS sea impredecible y, por tanto, no se permite.

Inicialización de la pila

Cuando se inicializa la zona de la pila, se cargan el registro (SS) de segmento de pila y el registro (SP) apuntador de pila. Se acostumbra designar una zona de la memoria como segmento de pila, para lo cual se carga SS con la localidad del "fondo" del segmento de pila.

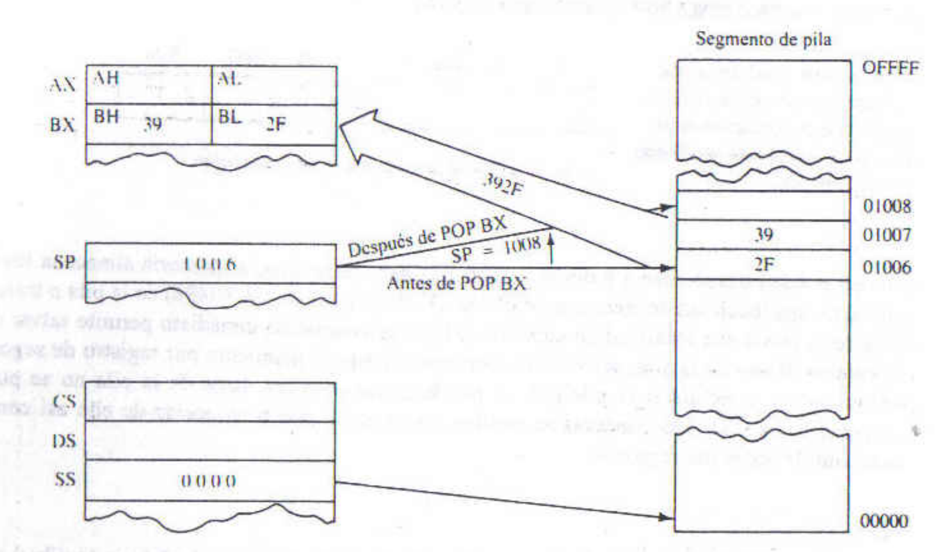


FIGURA 3–13 POP BX transfiere dos bytes de datos del segmento de pila y al registro BX.

Por ejemplo, si el segmento de pila se encuentra en las localidades 10000H-1FFFFH de la memoria, se carga SS con un 1000H. (Recuerde que se añade al extremo derecho del registro de segmento de pila un 0H para el direccionamiento en modo real.) Para inicializar la pila en la parte de este segmento de pila de 64K bytes, se carga el apuntador de pila (SP) con 0000H. En la figura 3-14 se muestra la forma en que este valor ocasiona que se transfieran los datos hacia la parte superior del segmento de pila con una instrucción PUSH CX. Recuerde que todos los segmentos son *cíclicos*, es decir que la localidad superior de un segmento es contigua a la localidad inferior del segmento.

TABLA 3–8 Las instrucciones POP

Simbólica	*Ejemplo*	*Funciones*
POP reg16	POP DI	Registro de 16 bits
POP reg32	PO EBX	Registro de 32 bits
POP mem16	POP WORD PTR[DI+2]	Dirección de 16 bits en la memoria
POP mem32	POP DATA3	Dirección de 32 bits en la memoria
POP seg	POP GS	Cualquier registro de segmento
POPA	POPA	Registros de 16 bits
POPAD	POPAD	Registros de 32 bits
POPF	POPF	Registro de bandera de 16 bits
POPFD	POPFD	Registro extendido de banderas de 32 bits

Nota: Se requieren 80386/80486 para trabajar con direcciones y registros de 32 bits.

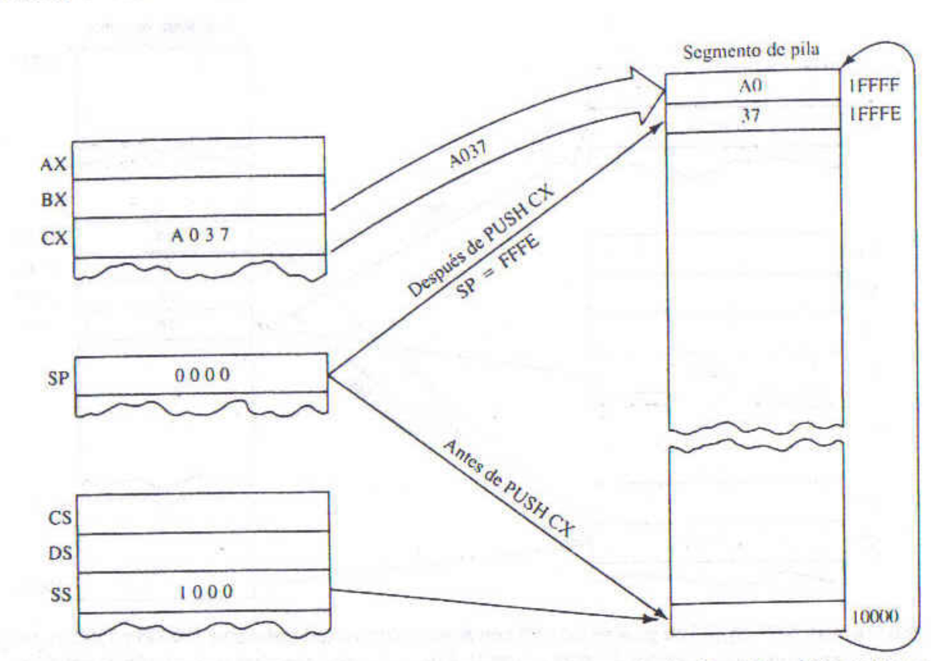


FIGURA 3–14 PUSH CX para ilustrar la naturaleza cíclica del segmento de pila. Se verá que SP empieza con 0000H y , después de decremento, termina con FFFEH.

EJEMPLO 3–1

```
0000                STACK_SEG   SEGMENTO DE PILA

0000  0100[                DW   100H DUP (?)
             ????
                 ]
0200                STACK_SEG   ENDS
```

En el lenguaje ensamblador, un segmento de pila se inicializa como se ilustra en el ejemplo 3-1. El primer enunciado determina el inicio del segmento de pila y el último determina el final del segmento de pila. El ensamblador y el programa ligador colocan la dirección correcta del segmento de pila en SS y la longitud del segmento (parte superior de la pila) en SP. No hay necesidad de cargar estos registros en el programa, salvo que se desee cambiar, por alguna razón, los valores iniciales.

Un método alterno para definir el segmento de pila se utiliza con uno de los modelos de memoria sólo en el ensamblador MASM (véase el apéndice A). En otros ensambladores no se utilizan modelos o, si se hace, no son exactamente los mismos que con el MASM. En este caso (véase el ejemplo 3-2) la declaración .STACK seguida por el número de bytes asignados en la pila, define el área de la pila, lo cual es idéntico al ejemplo 3-1. La declaración .STACK también inicializa a SS y SP.

Si no se especifica la pila con el empleo de uno u otro método, aparecerá una advertencia cuando se encadena el programa. Se puede pasar por alto la advertencia si el tamaño de la pila es de 128 bytes o menos. El sistema asigna en forma automática (por medio del DOS) una sección de 128 bytes de la memoria a la pila. Esta sección de memoria se ubica en el prefijo de segmento de programa, que está colocado al inicio de cada programa. Si se utiliza más memoria para la pila, se borrará información en el PSP que es de importancia crítica para el funcionamiento del programa y de la computadora.

EJEMPLO 3–2

```
.MODEL SMALL
.STACK 200H         ;inicializar tamaño de pila
```

3–3 CARGAR LA DIRECCION EFECTIVA

Hay cierto número de instrucciones para cargar la dirección efectiva (LEA) en el conjunto de instrucciones de los microprocesadores 8086-80486. La instrucción LEA carga cualquier registro de 16 bits con la dirección determinada por el modo de direccionamiento seleccionado para la instrucción. Las variantes LDS y LES cargan cualquier registro de 16 bits con la dirección de desplazamiento recuperada de una localidad en la memoria y, luego, cargan a DS o ES con una dirección de segmento recuperada de la memoria. En los microprocesadores 80386 y 80486, se suman LFS, LGS y LSS al conjunto de instrucciones y se puede seleccionar un registro de 32 bits para un desplazamiento de 32 bits de la memoria. En la tabla 3-9 se presentan las instrucciones para la carga de la dirección efectiva.

LEA

La instrucción LEA carga un registro de 16 bits con la dirección de desplazamiento de los datos especificada por el operando. Como se indica en el primer ejemplo de la tabla 3-9, la dirección NUMB del operando se carga en el registro AX, pero no su contenido.

TABLA 3–9 Las instrucciones para carga de la memoria efectiva

Simbólica	*Funciones*
LEA AX, NUMB	Se carga AX con la dirección de NUMB
LEA EAX, NUMB	Se carga EAX con la dirección de NUMB
LDS DI, LIST	Se cargan DI y SI con la dirección almacenada en LISTA
LDS EDI, LIST	Se cargan EDI y DS con la dirección almacenada en LISTA
LES BX, CAT	Se cargan BX y ES con la dirección almacenada en GAT
LFS DI, DATA1	Se cargan DI y FS con la dirección almacenada en DATO1
LGS SI, DATA5	Se cargan SI y GS con la dirección almacenada en DATO5
LSS SP, MEM	Se cargan SP y SS con la dirección almacenada en MEM

Al comparar LEA con MOV, se observa el siguiente efecto: LEA BX,[DI] carga la dirección de desplazamiento especificada por [DI] (contenido de DI) en el registro BX. Con MOV BX,[DI] se cargan en el registro BX los datos almacenados en la localidad de memoria direccionada por [DI].

En páginas anteriores se presentaron algunos ejemplos con el empleo del pseudooperador OFFSET. El *directivo* OFFSET efectúa la misma función que una instrucción LEA si el operando es un desplazamiento. Por ejemplo, MOV BX,OFFSET LISTA desempeña la misma función que LEA BX,LISTA. Ambas instrucciones cargan la dirección de desplazamiento de la localidad LISTA de la memoria en el registro BX.

Pero, ¿por qué está disponible la instrucción LEA si el directivo OFFSET efectúa la misma tarea? Primero, OFFSET sólo funciona con operandos sencillos como LISTA; no se puede emplear para un operando como [DI] , LISTA [SI] etcétera. El directivo OFFSET es más eficiente que la instrucción LEA para operandos sencillos. El microprocesador requiere más tiempo para ejecutar LEA BX,LISTA que MOV BX,OFFSET LISTA. En el 80286 se requieren por ejemplo, 3 ciclos de reloj para ejecutar la instrucción LEA BX,LISTA y sólo dos para ejecutar MOV BX,OFFSET LISTA. La razón por la cual MOV BX, OFFSET LISTA se ejecuta con más rapidez, es porque el ensamblador calcula la dirección de desplazamiento de LISTA, mientras que con la instrucción LEA el microprocesador efectúa el cálculo cuando ejecuta la instrucción. La instrucción MOV BX,OFFSET LISTA realmente se ensambla como una instrucción de transferencia inmediata y es más eficiente.

Supóngase que el microprocesador ejecuta una instrucción LEA BX,[DI] y que DI contiene 1000H. Ya que DI contiene la dirección de desplazamiento, el microprocesador transfiere una copia de DI a BX. La instrucción MOV BX,DI efectúa esta tarea en menos tiempo y, a menudo, se le prefiere a la instrucción LEA BX,[DI].

Otro ejemplo es LEA SI,[BX + DI]. Esta instrucción suma BX a DI y almacena la suma en el registro SI. La suma generada es una suma modular de 64K. Si BX = 1000H y DI = 2000H, la dirección del desplazamiento transferido a SI es 3000H. Si BX = 1000H y DI = FF00H, la dirección desplazada es 0F00H. Se debe tener en cuenta que el segundo resultado es una suma modular de 64K. (Una *suma modular de 64K*, no considera ningún acarreo del resultado de 16 bits.)

LDS, LES, LFS, LGS y LSS

Las instrucciones LDS, LES, LFS, LGS y LSS, cargan cualquier registro de 16 o de 32 bits con una dirección de desplazamiento y a los registros de segmento DS, ES, FS, GS o SS con una dirección de segmento. En estas instrucciones se utiliza cualquiera de los modos de direccionamiento de la memoria para accesar a una sección de la memoria de 32 o de 48 bits que contenga el segmento y la dirección de desplazamiento. En estas instrucciones no se puede emplear el modo de direccionamiento por registro (MOD = 11). Se debe tener en cuenta que las instrucciones LFS, LGS y LSS igual que los registros de 32 bits sólo están disponibles en los microprocesadores 80386 y 80486.

En la figura 3-15 se ilustra un ejemplo de la instrucción LDS BX,[DI] la cual transfiere el número de 32 bits direccionado por DI segmento de datos, hacia los registros BX y DS. Las instrucciones LDS, LES, LFDS, LGS y LAA obtienen una nueva dirección lejana en la memoria. La dirección del desplazamiento aparece primero, seguida por la dirección del segmento. Este formato se utiliza para almacenar todas las direcciones de 32 bits en la memoria.

Una dirección lejana en la memoria se puede almacenar con el ensamblador. Por ejemplo, la instrucción ADDR DD FAR PTR FROG almacena la dirección de desplazamiento y la del seg-

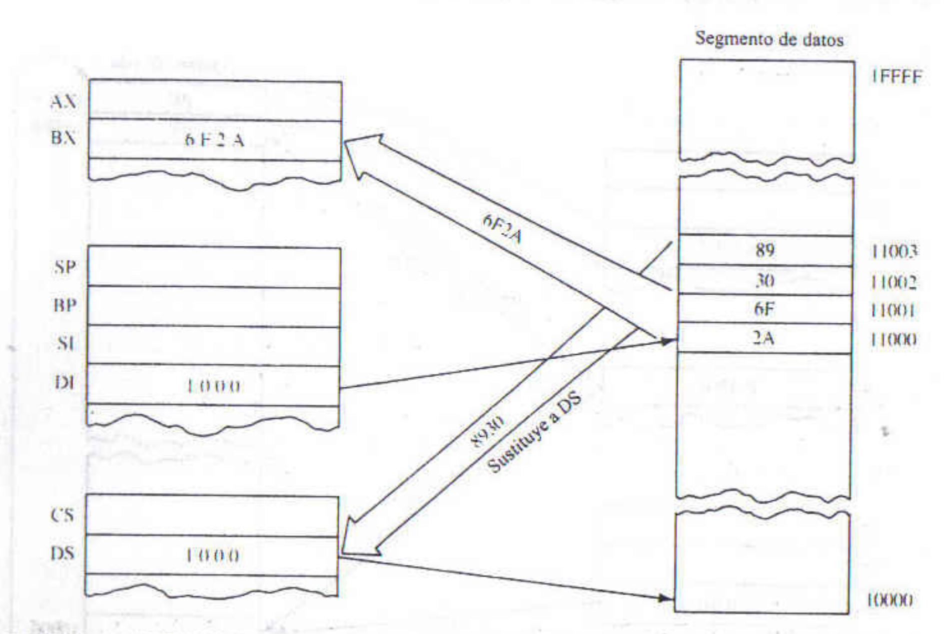


FIGURA 3–15 LDS BX,[DI] carga el registro BX con el contenido de las localidades 11000H y 11001H y al registro DS con el de las localidades 11002H y 11003H. Esta instrucción cambiará el segmento de datos de 10000H-1FFFFH, 89300H-992FFH.

mento (dirección lejana) de FROG en 32 bits de memoria en la localidad ADDR. El directivo DD ordena al ensamblador que almacene una *doble palabra* (número de 32 bits) en la dirección ADDR de la memoria.

En los 80386/80486 la instrucción LDS EBX,[DI] carga a EBX desde la sección de memoria de 4 bytes direccionada por DI en el segmento de datos. Después de este desplazamiento de 4 bytes hay una palabra que se carga en el registro DS. Se debe tener en cuenta que en vez de direccionar a una sección de memoria de 32 bits, los 80386/80486 direccionan una sección de 48 bits en la memoria siempre que se carga una dirección de desplazamiento de 32 bits en un registro de 32 bits.

3–4 TRANSFERENCIAS DE CADENAS DE DATOS

Hay cinco instrucciones para transferencia de cadenas de datos: LODS, STOS, MOVS, INS y OUTS. Cada una de ellas permite que las transferencias sean un solo byte o palabra o doble palabra o, si se repite, de un bloque de bytes, palabras o dobles palabras. Antes de describir las instrucciones para transferencia de cadenas, hay que entender el funcionamiento del bit D (dirección) del registro de banderas y de DI y SI, en lo tocante a las instrucciones para cadenas.

La bandera de dirección

La bandera (D) de dirección selecciona autoincremento (D =0) o autodecremento de los registros DI y SI durante las operaciones de cadenas. La bandera de dirección sólo se utiliza con las instrucciones para cadenas. La instrucción CLD *desactiva* la bandera D (D = 0) y la instrucción STD la *activa* (D = 1). Por tanto, la instrucción CLD selecciona el modo de autoincremento (D = 0) y la STD selecciona el modo de autodecremento (D = 1).

Siempre que una instrucción de cadena transfiere un byte, el contenido de DI y/o el de SI se incrementa o decrementa en 1. Si se transfiere una palabra, el contenido de DI y/o el de SI se incrementa o decrementa en 2. Las transferencias de doble palabra hacen que DI o SI incrementen o decrementen en 4. El incremento o decremento es sólo en los registros utilizados por la instrucción de cadena. Por ejemplo, la instrucción STOSB utiliza el registro DI para direccionar una localidad de la memoria. Cuando se ejecuta STOSB, sólo incrementa o decrementa SI sin afectar a Si. Lo mismo ocurre con la instrucción LODSB que utiliza el registro SI para direccionar datos en la memoria. LODSB sólo incrementa o decrementa a SI sin afectar a DI.

DI y SI

Durante la ejecución de una instrucción de cadena, los accesos a la memoria ocurren por uno o ambos de los registros DI y SI. La dirección de desplazamiento del DI accesa a los datos en el segmento extra para todos los datos que lo utilizan. La dirección de desplazamiento de SI accesa a los datos, en forma implícita, en el segmento de datos. La asignación del segmento SI se puede cambiar con un prefijo de cambio de segmento, descrito más adelante en este capítulo. La asignación del segmento de DI es *siempre* en el segmento adicional cuando se ejecuta una instrucción para cadena. Esta asignación no se puede cambiar. La razón de que un apuntador direccione a los datos en el segmento adicional y, el otro, al segmento de datos, es a fin de que la instrucción MOVS pueda transferir hasta 64K bytes de datos de un segmento a otro en la memoria.

LODS

La instrucción LODS carga AL, AX o EAX con datos almacenados en la dirección de desplazamiento dada por SI en el segmento de datos. (Se debe tener en cuenta que EAX sólo se puede emplear en los 80386 y 80486.) Después de cargar a AL con un byte, a AX con una palabra o a EAX con una doble palabra, el contenido de SI se incrementa si D = 0 o se decrementa si D = 1. Se suma o se resta un 1 en SI para una LODS de tamaño de byte, se suma o se resta un 2 para LODS de tamaño de palabra y se suma o se resta un 4 para LODS de tamaño de doble palabra.

En la tabla 3-10 se presentan las formas permitidas de la instrucción LODS. La instrucción LODSB (*carga un byte*) hace que se cargue un byte en AL; La instrucción LODSW (*carga una palabra*) hace que se cargue una palabra en AX; la instrucción LODSD (*carga una doble palabra*) hace que se cargue una doble palabra en EAX. Aunque es raro, como opción de LODSB, LODSW y LODSD, la instrucción LODS puede estar seguida por un operando de tamaño byte, palabra o doble palabra, para seleccionar un byte, palabra o doble palabra. A menudo se define a los operandos como bytes con DB, como palabras cuando tienen DW y como dobles palabras cuando tienen DD. La pseudooperando DB define uno o más bytes, DW *define palabra(s)* y DD *define doble(s) palabra(s)*.

TABLA 3-10 Formas de la instrucción LODS



Simbólica	*Funciones*
LODSB	AL = [SI]; SI = SI ± 1
LODSW	AX = [SI]; SI = SI ± 2
LODSD	EAX = [SI]; SI = SI ± 4
LODS LIST	AL = [SI]; SI = SI ± 1 (si LISTA es un byte)
LODS DATA1	AX = [SI]; SI = SI ± 2 (si DATO1 es una palabra)
LODS DATA4	EAX = [SI]; SI = SI ± 4 (si DATO4 es una doble palabra)

Nota: SI direcciona datos en el segmento de datos en forma implícita de LODS y las dobles palabras sólo se emplean en 80386/80486.

En la figura 3-16 se muestra el efecto de ejecutar la instrucción LODSW si la bandera D = 0, SI = 1000H y DS = 1000H. En este caso, un número de 16 bits almacenado en las localidades de memoria 11000H y 11001H, se carga AX. Debido a que D = 0 y que se trata de una transferencia de palabra, el contenido de SI se incrementa en 2, *después* de que se carga a AX con los datos de la memoria.

STOS

La instrucción STOS almacena AL, AX o EAX en el segmento extra en la localidad direcccionada por el registro DI.(Se debe tener en cuenta que en los 80386/80486 se utilizan EAX y dobles

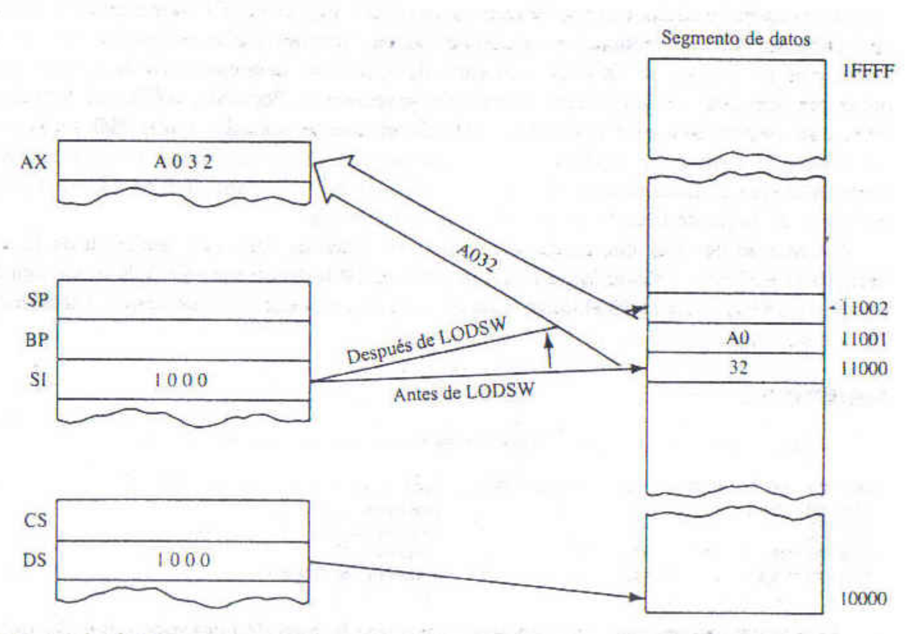


FIGURA 3-16 El efecto de ejecutar la instrucción LODSW si DS = 1000H, SI = 1000H, D = 0, 11000H = 32 y 11001H = A0.

TABLA 3–11 Formas de la instrucción STOS

Simbólica	*Funciones*
STOSB	[DI] = AL; DI = DI ± 1
STOSW	[DI] = AX; DI = DI ± 2
STOSD	[DI] = EAX; DI = DI ± 4
STOS LIST	[DI] = AL; DI = DI ± 1 (si LISTA es un byte)
STOS DATA1	[DI] = AX; DI = DI ± 2 (si DATO1 es una palabra)
STOS DATA4	[DI] = EAX; DI = DI ± 4 (si DATO4 es una doble palabra)

Nota: DI direcciona los datos en el segmento extra y las dobles palabras sólo se emplean en 80386/80486.

palabras.) En la tabla 3-11 se presentan todas las formas de la instrucción STOS; a ésta, igual que a la LODS se puede añadir una B, W o D para transferencias de byte, palabra o doble palabra. La instrucción STOSB (*almacena un byte*) almacena el byte en AL, en la localidad de memoria del segmento extra direccionada por DI. La instrucción STOSW (*almacena una palabra*) almacena AX en la localidad de memoria del segmento extra direccionada por DI. La instrucción STOSD (*almacena una doble palabra*) en la localidad de memoria en el segmento extra direccionada por DI. Después de almacenar el byte (AL), palabra (AX) o doble palabra (EAX), se incrementa o decrementa el contenido de DI.

STOS con REP. El *prefijo para repetir* (REP) se puede añadir a cualquier instrucción para transferencia de cadenas de datos excepto la instrucción LODS. El prefijo REP hace que CX se decremente en 1 cada vez que se ejecuta la instrucción de cadena. Después que se decrementa CX, se repite la instrucción en cadena. Si CX llega a un valor de 0, se termina la ejecución de la instrucción y el programa continúa con la siguiente instrucción en secuencia. Por tanto, si CX está cargado con un 100, y se ejecuta una instrucción REP STOSB, el microprocesador repite 100 veces en forma automática la instrucción STOSB. Dado que el registro DI se autoincrementa o autodecrementa después de que almacena cada dato, la instrucción almacena el contenido de AL en un bloque de memoria, en lugar de hacerlo en un solo byte de memoria.

Supóngase que hay que cargar con OOH, 10 bytes de datos, en una zona de la memoria llamada (BUFFER). Esto se logra con una serie de 10 instrucciones STOSB o con una STOSB con prefijo REP, si CX es 10 al inicio. Con el programa descrito en el ejemplo 3-3 se borra la zona de la memoria.

EJEMPLO 3–3

```
                ;inicializar a cero un bloque de memoria con el empleo de STOS

0000 C4 3E 05BC R   LES  DI,BUFFER     ;obtiene dirección de BUFFER
0004 B9 000A        MOV  CX,10         ;cargar contador
0007 FC             CLD                ;seleccionar autoincremento
0008 B0 00          MOV  AL,0          ;borrar AL
000A F3/ AA         REP  STOSB         ;borrar BUFFER
```

Esta breve secuencia de instrucciones direcciona la zona de la memoria con el empleo de la instrucción MOV DI,OFFSET BUFFER. Aunque BUFFER está en el segmento extra, el

ensamblador emplea una dirección de desplazamiento para direccionar la memoria. Se debe tener en cuenta que el prefijo REP va al principio de la instrucción STOSB en lenguaje ensamblador y en lenguaje de máquina. En el lenguaje de máquina, F3H es el prefijo REP y AAH es el código de STOSB.

Un método más rápido para borrar este bloque de memoria de 10 bytes es con la instrucción STOSW con un contador 5. En el ejemplo 3-4 se ilustra la misma tarea que en el ejemplo 3-3, excepto que el contador cambia a un 5 y se repite la instrucción STOSW en vez de la instrucción STOSB. Además, también se utiliza el registro AX en vez del registro AL.

EJEMPLO 3-4

```
        ;empieo de STOSW para borrar una memoria intermedia (BUFFER)

0000 C4 3E 05BC R   LES  DI,BUFFER     ;obtiene la dirección de BUFFER
0004 B9 0005        MOV  CX,5          ;cargar contador
0007 FC             CLD                ;seleccionar autoincremento automático
0008 B8 0000        MOV  AX,0          ;borrar AX
000B F3/ AB         REP  STOSW         ;borrar BUFFER
```

MOVS

La instrucción más útil para la transferencia de cadenas de datos es MOVS porque transfiere datos de una localidad a otra en la memoria. Esta es la única transferencia de *memoria a memoria* que se permite en los microprocesadores 8086-80486. La instrucción MOVS transfiere un byte, palabra o doble palabra desde la localidad del segmento de datos direccionada por SI, a la localidad en el segmento extra direccionada por DI. Al igual que en otras instrucciones de cadena, los apuntadores, se autoincrementan o autodecrementan según sea el valor de la bandera de dirección. En la tabla 3-12 se presentan todas las formas permisibles de la instrucción MOVS. Se debe tener en cuenta que sólo el segmento del operando fuente (SI), segmento de datos, se puede cambiar, por lo cual se puede utilizar otro segmento. El operando destino (DI) siempre está en el segmento extra.

Supóngase que el contenido de un arreglo de 100 bytes se debe transferir a otro de 100 bytes. La instrucción MOVSB (*transfiere un byte*), con prefijo de repetición REP, es ideal para esta tarea, como se ilustra en el ejemplo 3-5.

TABLA 3-12 Formas de la instrucción MOVS

Simbólica	*Funciones*
MOVSB	[DI] = [SI]; DI = DI ± 1; SI = SI ± 1 (byte transferido)
MOVSW	[DI] = [SI]; DI = DI ± 2; SI = SI ± 2 (palabra transferida)
MOVSD	[DI] = [SI]; DI = DI ± 4; SI = SI ± 4 (doble palabra transferida)
MOVS BYTE1, BYTE2	[DI] = [SI]; DI = DI ± 1; SI = SI ± 1 (si BYTE1 y BYTE2 son bytes)
MOVS WORD1, WORD2	[DI] = [SI]; DI = DI ± 2; SI = SI ± 2 (si WORD1 y WORD2 son palabras)
MOVS DWORD 1, DWORD2	[DI] = [SI]; DI = DI ± 4; SI = SI ± 4 (si DWORD1 y DWORD2 son dobles palabras)

EJEMPLO 3–5

```
                ;empleo de la instrucción MOVS

0000 C4 3E 05C0 R     LES  DI,LIST1       ;direccionar LISTA1
0004 C5 36 05C4 R     LDS  SI,LIST2       ;direccionar LISTA2
0008 FC               CLD                 ;borrar dirección
0009 B9 0064          MOV  CX,100         ;cargar contador
000C F3/ A4           REP  MOVSB          ;transferir 100 bytes
```

INS

La instrucción INS (*transfiere una cadena a la entrada*), que no está disponible en los microprocesadores 8086/8088, transfiere un byte, palabra o doble palabra de datos de un dispositivo de E/S hasta la localidad de memoria en el segmento extra direccionada por el registro DI. La *dirección de E/S* está contenida en el registro DX. Esta instrucción es útil para dar entrada a un bloque de datos, directamente a la memoria, desde un dispositivo E/S externo. Una aplicación transfiere datos de una unidad de disco a la memoria. A menudo las unidades de disco, se consideran interfaces de E/S en un sistema de computadora.

INS, igual que las instrucciones para cadenas ya citadas, tiene dos formas básicas. La instrucción INSB da entrada a datos de un dispositivo de E/S de 8 bits y los almacena en una localidad de memoria de tamaño-byte, direccionada por SI. La instrucción INSW da entrada a datos de E/S de 16 bits y los almacena en una localidad de memoria de tamaño-palabra. La instrucción INSD transfiere una doble palabra. Estas instrucciones se pueden repetir con el empleo del prefijo REP. Esto permite almacenar un bloque completo de datos de entrada, de un dispositivo E/S, en la memoria. En la tabla 3-13 se presentan las diversas formas de la instrucción INS.

En el ejemplo 3-6 se muestra un programa para transferir 50 bytes de un dispositivo E/S cuya dirección es 03ACH y almacenar esos datos en el arreglo de memoria LISTA. El programa asume que los datos están disponibles, en el dispositivo de E/S todo el tiempo. De otro modo, el programa debe comprobar si el dispositivo E/S está listo para transferir datos, impidiendo el empleo del prefijo REP.

TABLA 3–13 Formas de la instrucción INS

Simbólica	*Funciones*
INSB	[DI] = [DX]; DI = DI ± 1 (byte transferido)
INSW	[DI] = [DX]; DI = DI ± 2 (palabra transferida)
INSD	[DI] = [DX]; DI = DI ± 4 (doble palabra transferida)
INS LIST	[DI] = [DX]; DI = DI ± 1 (si LISTA es un byte)
INS DATA1	[DI] = [DX]; DI = DI ± 2 (si DATO1 es una palabra)
INS DATA4	[DI] = [DX]; DI = DI ± 4 (si DATO4 es una doble palabra)

Nota: [DX] indica que DX contiene la dirección del dispositivo de E/S. Estas instrucciones no están disponibles en los microprocesadores 8086/8088 y las dobles palabras sólo se emplean en 80386/80486.

EJEMPLO 3–6

```
                        ;empleo de REP INSB para transferir datos de entrada a un arreglo de memoria
                        ;
0000 BF 0000 R              MOV  DI,OFFSET LISTS          ;direcciona arreglo
0003 BA 03AC                MOV  DX,3ACH                  ;direcciona E/S
0006 FC                     CLD                           ;auto incrementa
0007 B9 0032                MOV  CX,50                    ;cargar conteo
000A F3/6C                  REP INSB                      ;dar entrada a los datos
```

OUTS

La instrucción OUTS (*transfiere una cadena a la salida*), que no está disponible en los microprocesadores 8086/8088, transfiere un byte, palabra o palabra de datos, de la localidad en la memoria en el segmento de datos direccionada por SI, un dispositivo E/S, direccionado por el registro DX como en el caso de la instrucción INS. En la tabla 3-14 se ilustran las diversas formas de la instrucción OUTS.

En el ejemplo 3-7 se ilustra un programa breve que transfiere datos de un arreglo de memoria en un dispositivo E/S. Este programa asume que el dispositivo E/S siempre está listo para recibir los datos.

EJEMPLO 3–7

```
                        ;empleo de REP OUTS para transferir datos de salida desde un arreglo de memoria
                        ;
0000 BE 0064 R              MOV  SI,OFFSET ARRAY          ;direccionar la matriz
0003 BA 03AC                MOV  DX,3ACH                  ;direccionar a E/S
0006 FC                     CLD                           ;auto incrementa
0007 B9 0064                MOV  CX,100                   ;carga contador
000A F3/6E                  REP OUTSB
```

3–5 INSTRUCCIONES DIVERSAS PARA TRANSFERENCIA DE DATOS

No hay que dejarse engañar por la palabra "diversas"; estas instrucciones se utilizan en los programas. Las instrucciones para transferencia de datos descritas en esta sección son XCHG, LAHF,

TABLA 3–14 Formas de la instrucción OUTS

Simbólica	*Funciones*
OUTSB	[DX] = [SI]; SI = SI ± 1 (byte transferido)
OUTSW	[DX] = [SI]; SI = SI ± 2 (palabra transferida)
OUTSD	[DX] = [SI]; SI = SI ± 4 (doble palabra transferida)
OUTS LIST	[DX] = [SI]; SI = SI ± 1 (si LISTA es un byte)
OUTS DATA1	[DX] = [SI]; SI = SI ± 2 (si DATO1 es una palabra)
OUTS DATA4	[DX] = [SI]; SI = SI ± 4 (si DATO4 es una doble palabra)

Nota: [DX] indica que DX contiene la dirección del dispositivo de E/S. Estas instrucciones no están disponibles en los microprocesadores 8086/8088 y las dobles palabras sólo se emplean en los 80386/80486.

SAHF, XLAT, IN, OUT, BSWAP, MOVSX y MOVZX. Debido a que las instrucciones diversas no se emplean tan a menudo como una instrucción MOV, se presentan todas en esta sección.

XCHG

La instrucción de intercambio (XCHG) intercambia el contenido de un registro con el contenido de cualquier otro registro o una localidad de la memoria. La instrucción XCHG no se puede ejecutar en registros de segmento ni con datos de memoria a memoria. Los intercambios son de tamaño byte, palabra o doble palabra (sólo en 80386/80486) y se utiliza cualquiera de los modos de direccionamiento descritos en el capítulo 2, excepto el direccionamiento inmediato. En la tabla 3-15 aparecen las formas de la instrucción XCHG.

La instrucción XCHG empleando el registro AX de 16 bits con otro registro de 16 bits, es el intercambio más eficiente. Esta instrucción es de un byte. Otras instrucciones XCHG requieren dos o más bytes, según sea el modo de direccionamiento seleccionado.

Cuando se utilizan un modo de direccionamiento de la memoria y el ensamblador, no importa cuál operando direcciona la memoria. La instrucción XCHG AL,[DI] es idéntica a la instrucción XCHG [DI],AL por lo menos en lo que toca al ensamblador.

Si se tiene el microprocesador 80386/80486, la instrucción XCHG puede intercambiar dobles palabras de datos. Por ejemplo, la instrucción XCHG EAX,EBX intercambia el contenido del registro EAX con el del registro EBX.

LAHF y SAHF

Las instrucciones LAHF y SAHF se utilizan muy poco porque fueron diseñadas como instrucciones *puente*. Estas instrucciones permitieron que el software del 8085 (uno de los primeros microprocesadores de 8 bits) se tradujera al software del 8086 con un programa de traducción. Debido a que cualquier software que requería traducción es probable que haya quedado concluido hace muchos años, estas instrucciones tienen escasa aplicación en la actualidad. La instrucción LAHF transfiere los 8 bits que están más a la derecha en el registro de bandera, al registro AH. La instrucción SAHF transfiere al registro AH hacia los 8 bits que están más a la derecha en el registro de bandera.

XLAT

La instrucción XLAT (*traduce*) carga en el registro AL a un número almacenado en una tabla en la memoria. Esta instrucción ejecuta la técnica de consulta directa a la tabla utilizada para convertir un código en otro. La instrucción XLAT, primero suma el contenido de AL con BX para

TABLA 3-15 Formas de la instrucción XCHG

Simbólica	*Funciones*
XCHG reg,reg	Intercambia registros de byte, palabra y doble palabra
XCHG reg,mem	Intercambia datos en la memoria byte, palabra o doble palabra, con datos del registro

Nota: Las dobles palabras sólo se emplean en 80386/80486.

formar una dirección de memoria en el segmento de datos. Luego transfiere el contenido de esa dirección a AL. Esta es la única instrucción que suma un número de 8 bits con un número de 16 bits.

Supóngase que una tabla de 7 segmentos para un dispositivo de visualización de diodos emisores de luz (LED) está almacenada en la memoria en la dirección TABLA. Luego, la instrucción XLAT "traduce" el número en BCD que hay en AL, a un código de 7 segmentos. En el ejemplo 3-8 se presenta un breve programa que convierte un código en BCD a un código de 7 segmentos. En la figura 3-17 se muestra un ejemplo del funcionamiento de este programa si TABLA = 1000H, DS = 1000H y el valor inicial de AL = 05H (un 5 en BCD). Después de la "traducción", el contenido de AL = 6DH.

EJEMPLO 3–8

```
                ;empleo de XLAT para convertir de BCD a código de 7 segmentos

0012 BB 000 R  MOV  BX,OFFSET TABLE    ;direccionar tabla
0015 D7        XLAT
```

IN y OUT

En la tabla 3-16 aparecen las formas de las instrucciones IN y OUT que efectúan operaciones de E/S. Se debe tener en cuenta que *sólo* el contenido de AL, AX o EAX se transfiere entre el

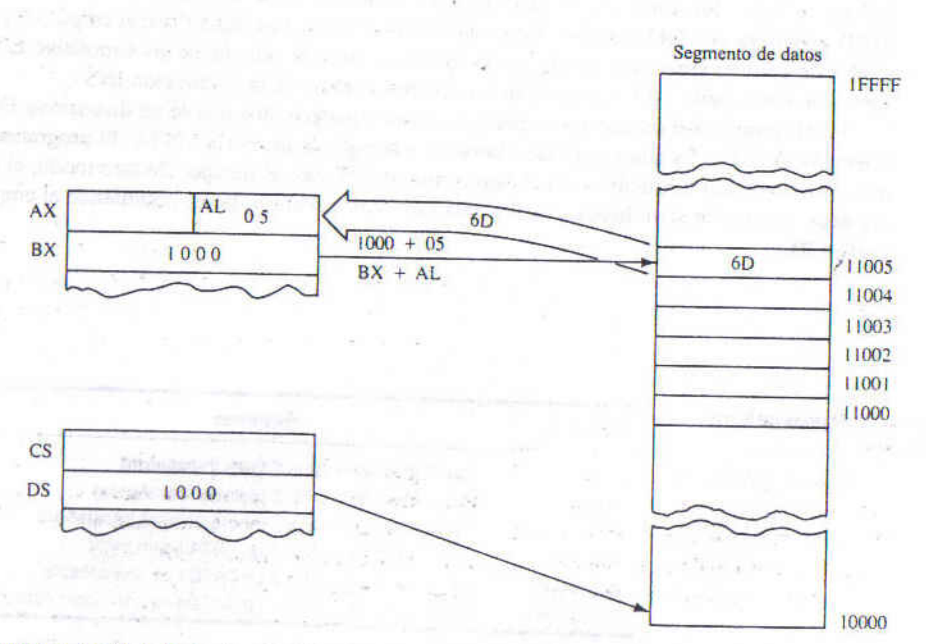


FIGURA 3–17 El efecto de ejecutar la instrucción XLAT en el punto justo antes de que el contenido 6DH de la localidad 11005H de la memoria se transfiera al registro AL.

TABLA 3–16 Instrucciones IN y OUT

Simbólicas	*Funciones*
IN AL,p8	Datos de 8 bits a AL, del puerto p8
IN AX,p8	Datos de 16 bits a AX del puerto p8
IN EAX,p8	Datos de 32 bits a EAX del puerto p8
IN AL,DX	Datos de 8 bits a AL del puerto DX
IN AX,DX	Datos de 16 bits a AX del puerto DX
IN EAX,DX	Datos de 32 bits a EAX del puerto DX
OUT p8,AL	Se envían datos de 8 bits de AL al puerto p8
OUT p8,AX	Se envían datos de 16 bits de AX al puerto p8
OUT p8,EAX	Se envían datos de 32 bits de EAX al puerto p8
OUT DX,AL	Se envían datos de 8 bits de AL al puerto DX
OUT DX,AX	Se envían datos de 16 bits de AX al puerto DX
OUT DX,EAX	Se envían datos de 32 bits de EAX al puerto DX

Nota: p8 = numero de puerto de 8 bits en E/S; DX = dirección de 16 bits del puerto en DX.

dispositivo E/S y el microprocesador. Una instrucción IN transfiere datos de un dispositivo de E/S a AL, AX o EAX; una instrucción OUT transfiere datos de AL, AX o EAX a un dispositivo de E/S. (Se debe tener en cuenta que sólo los 80386/80486 contienen EAX.)

Hay dos formas para la dirección del dispositivo (*puerto*) de E/S para IN y OUT: puerto fijo y puerto variable. El direccionamiento de puerto fijo permite la transferencia de datos entre AL, AX o EAX con el empleo de una dirección del puerto de E/S de 8 bits. Se le llama *direccionamiento de puerto fijo* porque el número del puerto sigue del código de la instrucción. A menudo, las instrucciones se almacenan en ROM. Una instrucción de puerto fijo tiene un número fijo de modo permanente por la naturaleza de la memoria de sólo lectura.

La dirección del puerto aparece en el canal de direcciones durante una operación de E/S. Para las instrucciones de 8 bits a un puerto fijo, la dirección de 8 bits del puerto se amplía con ceros a una dirección de 16 bits. Por ejemplo, si se ejecuta la instrucción IN AL,6AH, se da entrada a AL a los datos de la dirección 6AH del E/S. La dirección aparece como 006AH 16 bits en las terminales A0 a A15 del canal de direcciones; los bits A16 a A19 en el canal de direcciones (en 8086/8088), A16 a A23 (80286/80386SX), A16 a A24 (80386/L a 80386SLC) o A16 a A32 (80386-80486) son indefinidos para una instrucción IN o OUT.

El direccionamiento de puerto variable permite transferencias de datos entre AL,AX o EAX y una dirección de 16 bits del puerto. Se llama *direccionamiento de puerto variable* porque el número del puerto de E/S está cargado en el registro DX, y se puede cambiar (*variar*) durante la ejecución de un programa. La dirección de 16 bits del puerto E/S aparece en las terminales A0 a A15 del canal de direcciones. La PC de IBM utiliza una dirección de puerto de 16 bits para accesar el espacio de E/S, este espacio en la PC ocupa las localidades 0000H-03FFH. Por lo tanto, algunas tarjetas adaptadoras pueden usar direcciones de E/S arriba de 03FFH.

En la figura 3-18 se ilustra la ejecución de una instrucción OUT 19H,AX, que transfiere el contenido de AX al puerto 19H de E/S. Se debe tener en cuenta que el número de puerto de E/S aparece como 0019H en el canal de direcciones de 16 bits y que los datos de AX aparecen en el canal de datos de 16 bits de los microprocesadores 8086, 80286, 80386SX, 80386SL/80386SLC.

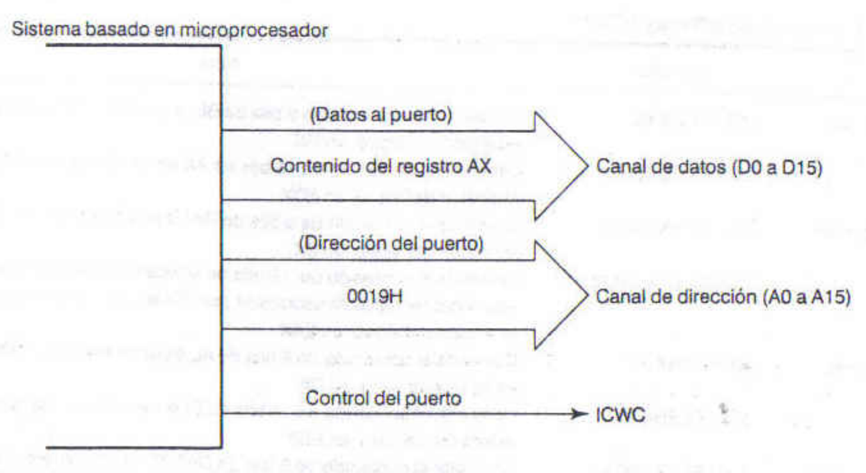


FIGURA 3–18 La ejecución de la instrucción OUT 19H, AX.

La señal de control del sistema $\overline{\text{IOWC}}$, (control de escritura de E/S) es un cero lógico para habilitar al dispositivo de E/S.

MOVSX y MOVZX

Las instrucciones MOVSX (transferir y extender el signo) y MOVZX (transferir y extender el cero) se encuentran en el conjunto de instrucciones en 80386 y 80486. Estas instrucciones transfieren datos y, al mismo tiempo, los extienden por signo o por cero. En la tabla 3-17 se ilustran estas instrucciones y algunos ejemplos de cada una.

Cuando se extiende un número con ceros, la parte más significativa se completa con ceros. Por ejemplo, si un 34H de 8 bits se extiende por ceros a un número de 16 bits, se vuelve 0034H. La extensión por cero se emplea a menudo para convertir números de 8 o de 16 bits sin signo en números de 16 o de 32 bits sin signo, con el empleo de la instrucción MOVZX.

Un número se extiende o amplía por signo cuando su bit de signo se copia y se pone en la parte más significativa. Por ejemplo, si un 84H de 8 bits se extiende por signo para que sea un número de 16 bits, se vuelve FF84H. El bit de signo de un 84H es un uno, el cual se copia y se pone en la parte más significativa del resultado extendido o ampliado por signo. La extensión por signo se usa más a menudo para convertir números de 8 o de 16 bits con signo en números de 16 o de 32 bits con signo, con el empleo de la instrucción MOVSX.

BSWAP

La instrucción para transferir o "canjear" bytes (BSWAP) sólo está disponible en el microprocesador 80486. Esta instrucción toma el contenido de cualquier registro de 32 bits y "canjea" el primer byte con el cuarto y el segundo con el tercero. Por ejemplo, la instrucción BSWAP EAX con EAX

TABLA 3–17 Instrucciones MOVSX y MOVZX

Simbólica	*Ejemplo*	*Nota*
MOVSX reg,reg	MOVSX CX,BL	Convierte el contenido de 8 bits de BL, a un número de 16 bits, por extensión del signo, en CX
	MOVSX ECX,AX	Convierte el contenido de 16 bits de AX en un número de 32 bits, por extensión del signo, en ECX
MOVSX reg,mem	MOVSX BX, DATA	Convierte el contenido de 8 bits de DATO en un número de 16 bits, por extensión del signo, en BX
	MOVSX EAX, [EDI]	Convierte el contenido de 16 bits de la localidad de memoria del segmento de datos direccionada por EDI en un número de 32 bits, por extensión del signo, en EAX
MOVZX reg,reg	MOVZX DX,AL	Convierte el contenido de 8 bits de AL en un número de 16 bits, por extensión del cero, en DX
	MOVZX EBP, DI	Convierte el contenido de 16 bits de DI en un número de 32 bits, por extensión del cero, en EBP
MOVZX reg,mem	MOVZX DX,DATA1	Convierte el contenido de 8 bits de DATO1 en un número de 16 bits, por extensión del cero, en DX
MOVZX reg,mem	MOVZX EAX,DATA2	Convierte el contenido de 16 bits de DATO2 en un número de 32 bits, por extensión del cero, en EAX

= 00112233H transfiere los bytes en EAX con lo cual el resultado es EAX = 33221100H. Se debe tener en cuenta que el orden de los cuatro bytes se invierte con esta instrucción.

3–6 PREFIJO PARA CAMBIO DE SEGMENTOS

El prefijo para cambio de segmentos se puede agregar a casi cualquier instrucción para 8086-80486 en cualquier modo de direccionamiento de la memoria, permite al programador cambiar el segmento implícito. El prefijo de cambio de segmento es un byte adicional que se añade al principio de una instrucción para seleccionar un registro de segmento alterno. Casi las únicas instrucciones en las que no se emplea un prefijo son JMP y CALL, en las cuales se debe emplear el registro del segmento de código para generar la dirección. El cambio de segmento también se utiliza para seleccionar los segmentos FS y GS en los microprocesadores 80386 y 80486.

Por ejemplo, la instrucción MOV AX,[DI] accesa datos del segmento de datos en forma implícita. Si así lo requiere el programa, se puede cambiar el segmento si se pone prefijo a la instrucción. Supóngase que los datos están en el segmento extra en lugar del segmento de datos. Si esta instrucción se cambia a MOV AX,ES:[DI] se direcciona al segmento extra.

En la tabla 3-18 se muestran algunas direcciones modificadas que direccionan a segmentos de la memoria que no son los implícitos. Cada vez que se emplea un prefijo de cambio de segmento en una instrucción, ésta se vuelve un byte más largo. Aunque no es ningún cambio grave en la longitud de la instrucción, sí aumenta su tiempo de ejecución. Por lo general, se limita el uso del prefijo de cambio de segmento y se permanece en los segmentos implícitos para escribir programas más breves y eficientes.

TABLA 3–18 Instrucciones que incluyen prefijos para cambio de segmento

Simbólica	*Segmento accesa*	*Segmento normal*
MOV AX,DS:[BP]	Segmento de datos	Segmento de pila
MOV AX,ES:[BP]	Segmento extra	Segmento de pila
MOV AX,SS:[DI]	Segmento de pila	Segmento de datos
MOV AX,CS:[SI]	Segmento de código	Segmento de datos
MOV AX,ES:LIST	Segmento extra	Segmento de datos
LODS ES:DATO	Segmento de datos	Segmento extra
MOV EAX,FS:DATO2	Segmento FS	Segmento de datos
MOV BL,GS:[ECX]	Segmento GS	Segmento de datos

Nota: Los segmentos FS y GS sólo se pueden utilizar en los 80386 y 80486.



3–7 DETALLES DEL ENSAMBLADOR

El ensamblador[1] para los microprocesadores 8086-80486 se puede utilizar en dos formas. En esta sección se describen ambos métodos y se explica cómo organizar el espacio de memoria del programa con el empleo del ensamblador. También se explica la finalidad y empleo de algunos de los directivos más importantes utilizados con este ensamblador. En el apéndice A aparecen detalles adicionales del ensamblador.

Directivos

Antes de describir el formato de un programa en lenguaje ensamblador, hay que conocer algunos detalles acerca de los directivos (pseudooperaciones) que controlan el ensamblador. En la tabla 3-19 aparecen algunos directivos comunes en lenguaje ensamblador. Los directivos indican cómo procesará el ensamblador un operando o una sección del programa. Algunos directivos generan y almacenan información en la memoria; otros, no. El directivo DB (define un byte) almacena bytes de datos en la memoria; el directivo BYTE PTR nunca almacena datos ya que indica el tamaño de las referencias a datos con un apuntador o un registro índice.

Se debe tener en cuenta que el ensamblador en forma implícita sólo acepta instrucciones para el 8086/8088 salvo que el software está precedido por el directivo .286 o .286P o uno de los otros conmutadores para selección del microprocesador. El directivo .286 ordena al ensamblador que utilice el conjunto de instrucciones del 80286 en el modo real; el directivo .286P ordena utilizar el conjunto de instrucciones del 80286 en modo protegido.

Almacenamiento de datos en un segmento de memoria. Los directivos DB (define un byte), DW (define una palabra) y DD (define doble palabra) se utilizan más a menudo con los 8086-80486 para definir y almacenar datos en la memoria. Si el sistema tiene un coprocesador numérico, también son usuales los directivos DQ (define una cuádruple palabra) y DT (define diez bytes).

[1] El ensamblador citado en el texto es el macroensamblador de Microsoft, llamado MASM.

TABLA 3–19 Directivos comunes del ensamblador

Directivo	*Función*
.286	Selecciona el conjunto de instrucciones para 80286
.286P	Selecciona el conjunto de instrucciones en modo protegido para 80286
.386	Selecciona el conjunto de instrucciones para 80386
.386P	Selecciona el conjunto de instrucciones en modo protegido para 80386
.486	Selecciona el conjunto de instrucciones para 80486
.486P	Selecciona el conjunto de instrucciones en modo protegido para 80486
.287	Selecciona el coprocesador numérico 80287
.387	Selecciona el coprocesador numérico 80387
ALIGN 2	Inicia los datos en un segmento con límites de palabra o de doble palabra
ASSUME	Indica los nombres de cada segmento al ensamblador; no carga los registros de segmento
AT	Indica la dirección física de segmento que se emplea con el enunciado SEGMENT
BYTE	Indica un operando de tamaño de byte como BYTE PTR o THIS BYTE
DB	Define uno o más bytes (8 bits)
DD	Define palabra o dobles palabras (32 bits)
DQ	Define palabra o cuádruples palabras (64 bits)
DT	Define 10 bytes (80 bits)
DUP	Genera duplicados de caracteres o números
DW	Define palabra o palabras (16 bits)
DWORD	Indica un operando de tamaño de doble palabra como en THIS DWORD
END	Indica el final del programa
ENDM	Indica el final de una secuencia de macro
ENDP	Indica el final de un procedimiento
ENDS	Indica el final de un segmento
EQU	Iguala los datos con los de una etiqueta
FAR	Especifica una dirección lejana como en JMP FAR PTR LISTAS
MACRO	Define el nombre, parámetros e inicio de un macro
NEAR	Especifica una dirección cercana como en JMP NEAR PTR AYUDA
OFFSET	Especifica una dirección de desplazamiento
ORG	Inicializa el origen dentro de un segmento
PROC	Define el inicio de un procedimiento
PTR	Indica un apuntador a la memoria
SEGMENT	Define el comienzo de un segmento de memoria
STACK	Indica que un segmento es segmento de pila
STRUC	Define el comienzo de una estructura de datos
THIS	Se emplea con EQU para establecer una etiqueta de un byte, palabra o doble palabra
USES	Un directivo de la versión 6.0 de MASM que salva en forma automática los registros utilizados en un procedimiento
USE16	Ordena al ensamblador que utilice el modo de instrucción y tamaños de datos de 16 bits para 80386 y 80486
USE32	Ordena el ensamblador que utilice el modo de instrucción y tamaños de datos de 32 bits para los 80386 y 80486
WORD	Actúa como operando palabra como en WORD PTR o THIS WORD

Nota: La mayor parte de estos directivos funcionan con la mayoría de las versiones del ensamblador.

En estos directivos se utiliza una etiqueta para identificar una localidad en la memoria y el directivo para indicar su tamaño.

EJEMPLO 3–9

```
                            empleo de DB, DW y DD
                            ;
0000                        LIST_SEG     SEGMENT

0000 01 02 03               DATA_ONE     DB      1,2,3        ;define byte
0003 45                                  DB      45H          ;hexadecimal
0004 41                                  DB      'A'          ;ASCII
0005 F0                                  DB      11110000B    ;binario
0006 000C 000D              DATA_TWO     DW      12,13        ;define palabras
000A 0200                                DW      LIST1        ;simbólico
000C 2345                                DW      2345H        ;hexadecimal
000E 00000300               DATA_THREE   DD      300H         ;hexadecimal
0012 4007DF3B                            DD      2.123        ;real
0016 544269E1                            DD      3.34E+12     ;real
001A 00                     LISTA        DB      ?            ;reservar 1 byte
001B 000A[                  LISTB        DB      10 DUP (?)   ;reservar 10 bytes
                  ??
                            ]
0025 00                                  ALIGN   2            ;inicializa limite de palabra

0026 0100[                  LISTC        DW      100H DUP (0)
                  0000
                            ]
0226 0016[                  LIST_NINE    DD      22 DUP (?)
                  ????????
                            ]
0027E 0064[                 SIXES        DB      100 DUP (6)
                  06
                            ]

02E2                        LIST_SEG     ENDS
                                         END
```

En el ejemplo 3-9 se muestra un segmento de memoria que contiene varios de directivos para la definición de datos. El primer enunciado indica el comienzo del segmento y su nombre simbólico. El ultimo enunciado del segmento contiene el directivo ENDS que indica el final del segmento. El nombre del segmento (LISTA_SEG) puede ser cualquier cosa que desee llamarle el programador.

En este ejemplo, se muestran diversas formas de almacenamiento de datos para bytes en DATO_UNO. Se puede definir más de un byte en una línea en código binario, hexadecimal, decimal o ASCII. La etiqueta DATO_DOS muestra cómo se almacenan diversas formas de datos palabra. Las dobles palabras se almacenan en DATO_TRES e incluyen números reales, de punto flotante, de precisión simple.

Se puede reservar memoria para utilizarla en el futuro si se utiliza el signo ? como operando para un directivo DB, DW o DD. Cuando se utiliza ? en lugar de un valor numérico o ASCII, el ensamblador pone aparte o reserva una localidad y no la inicializa a ningún valor. El directivo DUP (duplicar) produce un arreglo como se ilustra en diversas formas en el ejemplo 3.9. Un 10 DUP (?) reserva 10 localidades de memoria, pero no almacena ningún valor específico en ningu-

na de las 10. Si aparece un número entre los paréntesis () de la declaración DUP, el ensamblador inicializa con datos la sección de memoria reservada.

El directivo ALIGN utilizado en este ejemplo, comprueba que los arreglos de memoria estén almacenados dentro de los límites de palabras. Una ALIGN 2 coloca datos en límites de palabras para el microprocesador; una ALIGN 4 los pone en los límites de doble palabra para un 80386 u 80486 y datos de doble palabra. Es importante que los datos de tamaño palabra se pongan en los límites de palabra y, los de tamaño de doble palabra, en los límites de ésta. Si no se hace así, el microprocesador tarda más tiempo del necesario para accesar estos datos. Una palabra almacenada en una localidad impar en la memoria requiere el doble de tiempo para accesarla que una palabra almacenada en una localidad de memoria par.

EQU y THIS. El directivo EQU iguala un valor numérico, ASCII o etiqueta con otra etiqueta. El directivo EQU hace que el programa esté más claro y simplifica la depuración. En el ejemplo 3-10 se muestran algunos directivos EQU y algunas instrucciones para mostrar cómo funcionan en un programa.

EJEMPLO 3–10

```
                    ;empleo de EQU (igualar)
                    ;
= 000A              TEN         EQU  10
= 0009              NINE        EQU  9

0000 B0 0A                      MOV  AL,TEN
0002 04 09                      ADD  AL,NINE
```

El directivo THIS aparece siempre como THIS BYTE, THIS WORD o THIS DWORD. En algunos casos, los datos se deben llamar un byte y una palabra. El ensamblador sólo puede asignar una dirección byte o palabra a una etiqueta. Para asignar una etiqueta byte a una palabra, se utiliza el programa del ejemplo 3-11.

EJEMPLO 3–11

```
                    ;empleo de ORG y THIS
                    ;
0000                DATA_SEG          SEGMENT

0100                          ORG     100H

= 0100              DATA1     EQU     THIS BYTE
0100 0000           DATA2     DW  ?

0102                DATA_SEG      ENDS

0000                CODE_SEG      SEGMENT 'CODE'

                              ASSUME CS:CODE_SEG,DS:DATA_SEG

0000 8A 1E 0100 R             MOV  BL,DATA1
0004 A1 0100 R                MOV  AX,DATA2
0007 8A 3E 0101 R             MOV  BH,DATA1+1

000B                CODE_SEG      ENDS
```

En este ejemplo se ilustra también la forma en que el enunciado ORG (origen) cambia la dirección inicial de los datos en el segmento de datos, a la localidad 100H. El enunciado ASSUME le da al ensamblador los nombres que se han seleccionado para los segmentos de código, datos, extra y de pila. Al no tener este enunciado, el ensamblador no asume nada y utiliza un prefijo en todas las instrucciones que direccionan datos en la memoria.

PROC y ENDP. Los directivos PROC y ENDP indican el principio y el final de un procedimiento (*subrutina*). Estos directivos hacen que haya estructura, porque el procedimiento está definido con claridad. Ambos directivos requieren una etiqueta para indicar el nombre de la subrutina. PROC, que indica el comienzo de un procedimiento debe estar seguida de NEAR (Cercano) o FAR (Lejano). Un procedimiento CERCANO se encuentra en el mismo segmento de código que el programa. Un procedimiento LEJANO puede estar en cualquier localidad del sistema de memoria. A menudo a los procedimientos cercanos se les llama locales y a los lejanos, globales. El término global denota un procedimiento que puede utilizar cualquier programa, mientras que local es un procedimiento que sólo se utiliza en el programa que está en curso.

EJEMPLO 3–12

```
                    ;ejemplo para sumar BX, CX y DX y almacenar
                    ;la suma en AX
                    ;
0000                ADDEM       PROC  FAR

0000 03 D9                      ADD   BX,CX
0002 03 DA                      ADD   BX,DX
0004 8B C3                      MOV   AX,BX
0006 CB                         RET

0007                ADDEM       ENDP
```

En el ejemplo 3-12 se muestra un procedimiento que suma BX, CX y DX y almacena la suma en el registro AX. Aunque este procedimiento es breve, y quizá no tenga tanta utilidad, ilustra el empleo de los directivos PROC y ENDP para delinear el procedimiento.

Si se tiene la versión 6.0 del programa ensamblador MASM de Microsoft, el directivo PROC puede especificar y salvar en forma automática cualesquiera de los registros utilizados. El enunciado USES indica cuáles registros se emplean en el procedimiento a fin de que el ensamblador los pueda salvar en forma automática, antes de que empiece el procedimiento y los recupere antes de que termine el mismo, con la instrucción RET. Por ejemplo, el enunciado ADDS PROC USES AX BX CX, salva en forma automática AX, BX y CX en la pila antes de que empiece el procedimiento, los recupera antes que se ejecute la instrucción RET al final del procedimiento. En el ejemplo 3-13 se ilustra un procedimiento escrito con el MASM 6.0 que muestra el enunciado USES. Se debe tener en cuenta que los registros en esa lista no están separados por una coma y que las instrucciones PUSH y POP no se exhiben en el listado de procedimiento. El enunciado USES no aparece en ningún otro lugar en este libro, por lo cual se puede mantener la compatibilidad con el MASM versión 5.10.

EJEMPLO 3–13

```
                    ;procedimiento que incluye la directiva USES para salvar
                    ;BX, CX y DX en la pila.
                    ;
```

```
0000                ADDS    PROC CERCANO USES BX CX, DX

0003 03 D8                  ADD  BX,AX
0005 03 CB                  ADD  CX,BX
0007 03 D1                  ADD  DX,CX
0009 8B C2                  MOV  AX,DX
                            RET

000F                ADDS    ENDP
```

Organización de la memoria

En el ensamblador se utilizan dos formatos básicos para desarrollar programación. En un método se utilizan modelos y, en el otro, definiciones del segmento completo. Los modelos de memoria que se describen en esta sección, son exclusivos del programa MASM para ensamblador. En el ensamblador TASM también se emplean modelos de memoria, pero son diferentes de los modelos MASM. Las definiciones de segmento completo son comunes para casi todos los ensambladores, incluso en el *ensamblador Intel*, y casi siempre se utilizan para desarrollar programas. Los modelos son más fáciles de usar, pero las definiciones de segmento completo ofrecen mayor control de la tarea del lenguaje ensamblador y se recomiendan y mencionan en otros lugares en este libro. Se utilizan segmentos completos porque se aplican a todos los ensambladores, pero los modelos no.

Modelos. Hay muchos modelos que se pueden utilizar con el ensamblador MASM, desde los muy pequeños hasta los muy grandes. En el apéndice A hay una tabla en que se mencionan todos los modelos disponibles para utilizarlos con el ensamblador. Para designar un modelo, se utiliza el enunciado MODEL seguido por el tamaño del sistema de memoria. El modelo muy pequeño requiere que el programa y los datos quepan en un segmento de memoria de 64K bytes y sean útiles para los programas pequeños. En el modelo pequeño, se requiere que sólo se utilice un segmento de datos y un segmento de código, para un total de 128K bytes de memoria. Hay otros modelos disponibles hasta llegar al muy grande.

EJEMPLO 3–14

```
                            .MODELO SMALL

                            .STACK 100H        ;definir pila

                            .DATA              ;definir datos

0000 0064[          LISTA   DB     100 DUP (?)
            ??
                ]

0064 0064[          LISTB   DB     100 DUP (?)
            ??
                ]

                            .CODE              ;definir código

0000 B8-R           HERE:   MOV  AX,@DATA      ;cargar ES, DS
0003 8E C0                  MOV  EX,AX
0005 8E D8                  MOV  DS,AX

0007 FC                     CLD                ;transferir datos
```

```
0008 BE 0000 R          MOV   SI,OFFSETLISTA
000B BF 0064 R          MOV   DI,OFFSETLISTB
000E B9 0064            MOV   CX,100
0011 F3/A4              REP MOVSB

0013                    .EXIT 0                    ;regresa a DOS

                        END  HERE
```

En el ejemplo 3-14 se ilustra cómo el enunciado MODEL define los parámetros de un programa corto que copia el contenido de un bloque de memoria de 100 bytes (LISTA) y lo pone en un segundo bloque de memoria de 100 bytes (LISTB).

EJEMPLO 3–15

```
0000               STACK_SEG      SEGMENT STACK

0000 0100[                        DW      100H DUP (?)
          ????
           ]

0200               STACK_SEG      ENDS

0000               DATA_SEG       SEGMENT 'DATA'

0000 0064[                  LISTA DB      100 DUP (?)
          ??
              ]

0064 0064[                  LISTB DB      100 DUP (?)
          ??

              ]

00C8               DATA_SEG       ENDS

0000               CODE_SEG       SEGMENT 'CODE'

                            ASSUME CS:CODE_SEG,DS:DATA_SEG,SS:STACK_SEG

0000                  PRINCIPAL  PROC     FAR

0000 B8–R                        MOV      AX,DATA_SEG    ;cargar DS y ES
0003 8E C0                       MOV      ES,AX
0005 8E D8                       MOV      DS,AX

0007 FC                          CLD                     ;transferir datos
0008 BE 0000 R                   MOV      SI,OFFSET LISTA
000B BF 0064 R                   MOV      DI,OFFSET LISTB
000E B9 0064                     MOV      CX,100
0011 F3/A4                    REP MOVSB

0013 B4 4C                    MOV     AH,4CH          ;regresa a DOS
0015 CD 21                    INT     21H

0017                    MAIN  ENDP

0017              CODE_SEG    ENDS

                              END     MAIN
```

Definiciones de segmento completo. En el ejemplo 3-15 se ilustra el mismo programa con el empleo de definiciones de segmentos completos. El programa parece ser más largo pero está mejor estructurado que en el método de modelos para inicializar un programa. El primer segmento definido es STACK_SEG (segmento de pila) que se define con claridad con los directivos SEGMENT y ENDS. Dentro de estos directivos, un DW 100 DUP (?) reserva 100H palabras para el segmento de pila. Debido a que STACK aparece junto a SEGMENT, el ensamblador y el ligador cargan en forma automática el registro (SS) del segmento de pila y el apuntador (SP) de la pila.

A continuación, los datos se definen en el DATA_SEG. En este caso, aparecen dos arreglos de datos como LISTA y LISTB. Cada uno tiene 100 bytes de espacio para el programa. Los nombres de los segmentos en este programa se pueden cambiar por cualquier otro. Se incluye el nombre grupal 'DATA' para permitir el empleo eficaz del programa CODEVIEW de Microsoft para depurar el software. Si no se coloca el nombre grupal en un programa, de todos modos se puede emplear CODEVIEW para depurar el programa, pero no se podrá depurar en su forma simbólica. Otros nombres grupales, 'STACK' 'CODE', etcétera, aparecen en el apéndice A.

El CODE_SEG está organizado como un procedimiento lejano porque la mayor parte de la programación está orientada al procedimiento. Antes que comience el programa, el segmento de código contiene el enunciado ASSUME (asume), que le indica al ensamblador y al ligador el nombre utilizado para el segmento de código. (CS) es CODE_SEG y para el segmento de pila es STACK_SEG. Además, se verá que se ha incluido el nombre grupal 'CODE' para el segmento de código. En el apéndice A aparecen otros nombres grupales con los modelos.

Una vez que el programa carga el registro del segmento extra y el registro del segmento de datos con la localidad del segmento de datos, transfiere 100 bytes desde LISTA hasta LISTB. Después hay una secuencia de dos instrucciones que devuelven el control al DOS (el sistema operativo del disco).

El último enunciado en el programa es END MAIN. El enunciado END indica el final del programa y la localidad de la primera instrucción ejecutada. En este caso, se desea que la máquina ejecute el procedimiento principal (MAIN) por lo cual el directivo END va seguida por una etiqueta. En un archivo que está encadenado con otro archivo, no hay etiqueta en el directivo END.

En los microprocesadores 80386 y 80486 se encuentra un directivo adicional añadido al segmento de código: el directivo USE16 o USE32 indica al ensamblador que utilice los modos de instrucción de 16 o de 32 bits para el microprocesador. En la programación desarrollada bajo operativo DOS se debe utilizar USE16 para que el programa funcione correctamente en los 80386 y 80486. En realidad, cualquier programa destinado a ejecutarse en modo real, debe incluir el directivo USE16. En el ejemplo 3-16 se ilustra la forma en que se forma el mismo programa del ejemplo 3-25 para los microprocesadores 80386 y 80486.

EJEMPLO 3–16

```
                                .386                                ;seleccionar el 80386
0000                            STACK_SEG    SEGMENT STACK

0000 0100[                                   DW       100H DUP (?)
                    ????
                      ]
```

```
0200                    STACK_SEG    ENDS

0000                    DATA_SEG     SEGMENT  'DATA'

0000 0064[                    LISTA DB     100 DUP (?)
                ??
                  ]

0064 0064[                    LISTB DB     100 DUP (?)
                ??
                  ]

00C8            DATA_SEG              ENDS

0000            CODE_SEG              SEGMENT USE16  'CODE'

                              ASSUME CS:CODE_SEG,DS:DATA_SEG,SS:STACK_SEG

0000                          MAIN PROC     FAR

0000 B8--R                        MOV      AX,DATA_SEG      ;cargar DS y ES
0003 8E C0                        MOV      ES,AX
0005 8E D8                        MOV      DS,AX

0007 FC                           CLD                       ;transferir datos
0008 BE 0000 R                    MOV      SI,OFFSET LISTA
000B BF 0064 R                    MOV      DI,OFFSET LISTB
000E B9 0064                      MOV      CX,100
0011 F3/A4                        REP MOVSB

0013 B4 4C                        MOV      AH,4CH           ;regresa a DOS
0015 CD 21                        INT      21H

0017                          MAIN  ENDP

0017            CODE_SEG          ENDS

                                  END      MAIN
```

Un programa de muestra

EJEMPLO 3-17

```
             ;program that reads a key and displays key
             ;an @ key ends the program
             ;
0000         CODE_SEG    SEGMENT  'CODE'

                      ASSUME   CS:CODE_SEG
0000               PRINCIPAL  PROC    FAR

0000 B4 06                    MOV     AH,6        ;¿hay tecla oprimida?
0002 B2 FF                    MOV     DL,0FFH
0004 CD 21                    INT     21H
0006 74 F8                    JE      MAIN        ;no hay tecla oprimida

0008 3C 40                    CMP     AL,'@'      ;probar si es @
000A 74 08                    JE      MAIN1       ;si hay @
```

```
000C B4 06              MOV   AH,6      ;exhibir tecla oprimida
000E 8A D0              MOV   DL,AL
0010 CD 21              INT   21H
0012 EB EC              JMP   MAIN      ;repetir
0014         MAIN1:
0014 B4 4C              MOV   AH,4CH    ;regresa a DOS
0016 CD 21              INT   21H

0018               MAIN ENDP

0018         CODE_SEG   ENDS

                        END   MAIN
```

En el ejemplo 3-17 se presenta un programa de muestra que lee un carácter del teclado y lo muestra en el monitor. Aunque el programa es muy sencillo, sirve para ilustrar un programa completo útil y que funciona en cualquier computadora personal en que se emplee el DOS, desde el primer sistema basado en el 8088, hasta el sistema más reciente basado en el 80486. En este programa también se ilustra el empleo de algunas llamadas a las funciones del DOS. En el apéndice A aparecen las llamadas a las funciones del DOS con sus parámetros. Las llamadas a las funciones del BIOS permiten el empleo del teclado, impresora, unidades de disco y todo lo demás que haya disponible en el sistema de la computadora.

En este programa de ejemplo, sólo se utiliza un segmento de código porque no hay datos. Debería aparecer un segmento de pila, pero se ha dejado fuera porque el DOS asigna en forma automática una pila de 256 bytes para todos los programas. La única ocasión en que se emplea la pila en este ejemplo, es para las instrucciones INT 21H que llaman a un procedimiento del DOS. Se debe tener en cuenta que cuando este programa se encadena, el ligador envía una advertencia de que no está presente ningún segmento de pila. En este ejemplo, se puede pasar por alto esta advertencia, porque la pila es de menos de 256 bytes.

En el programa se utilizan las funciones 06H y 4CH de DOS. El número de la función se coloca en AH antes de que se ejecute la INT 21H. La función 06H lee el teclado si DL = 0FFH o exhibe el contenido de ASCII de DL si no es 0FFH. Al verlo de cerca, la primera sección del programa carga a 06H a AH y 0FFH a DL, por lo cual se lee la tecla que se oprima en el teclado. La INT 21H prueba el teclado y si no se ha oprimido ninguna tecla, retorna un igual. La instrucción JE prueba la condición de igual (con la bandera de cero) y brinca a MAIN si no se ha oprimido ninguna tecla.

Cuando se oprime una tecla, el programa continúa al siguiente paso, en el cual se compara el contenido de AL con un símbolo @. Al retornar de la INT 21H, el carácter de ASCII de la tecla oprimida se encuentra en AL. En este programa, si se teclea un símbolo @, se termina el programa. Si no se teclea un símbolo @, el programa exhibe el carácter tecleado en el monitor con la siguiente instrucción INT 21H.

La segunda instrucción INT 21H transfiere al carácter ASCII a OL, para poder exhibirlo en el monitor. Después de exhibir el carácter, se ejecuta un JMP. Esto hace que el programa continúe en MAIN en donde repite con la lectura de una tecla oprimida.

Si se teclea el símbolo @, el programa va a la etiqueta MAIN1 en donde ejecuta la función 4CH del DOS. Esto hace que el programa retorne a la indicación (A>) a fin de poder usar la computadora para otras tareas.

En el apéndice A y en algunos de los capítulos siguientes aparece más información acerca del ensamblador y su aplicación. El apéndice incluye un panorama completo del ensamblador, ligador y las funciones del DOS. También incluye una lista de las funciones del BIOS (*sistema básico de*

E/S). La información que se presenta en los capítulos posteriores aclara cómo se puede utilizar el ensamblador para ciertas tareas que requieren diferentes grados de conocimiento.

3–8 RESUMEN

1. Las instrucciones para transferencia de datos los transfieren entre registros, un registro y memoria, un registro y la pila, memoria y la pila, el acumulador y E/S y entre el registro de las banderas y la pila.
2. Las instrucciones para transferencia de datos incluyen: MOV, PUSH, POP, XCHG, XLAT, IN, OUT, LEA, LDS, LES, LAHF, SAHF; las instrucciones para cadenas: LODS, STOS, MOVS, INS y OUTS.
3. El primer byte de una instrucción contiene el código de operación, que especifica la operación efectuada por el microprocesador. El código puede tener un prefijo de cambio de segmento en algunas formas de instrucciones.
4. El bit D, que se encuentra en muchas instrucciones, selecciona el sentido del flujo de datos. Si D = 1, los datos fluyen desde el campo REG hasta el campo R/M de la instrucción. Si D =0, los datos fluyen desde el campo R/M hasta el campo REG.
5. El bit W que se encuentra en muchas instrucciones, selecciona el tamaño de la transferencia de datos. Si W = 0, los datos son de tamaño de byte y si W = 1, los datos son de tamaño de palabra. En los microprocesadores 80386 y 80486, W = 1 también puede especificar un registro de 32 bits.
6. MOD selecciona el modo de funcionamiento del campo R/M de una instrucción en lenguaje de máquina. Si MOD = 00, no hay desplazamiento, si es 01, el desplazamiento es de 8 bits con signo extendido, si es 10, el desplazamiento es de 16 bits y si es 11, se utiliza un registro en lugar de una localidad en la memoria. En los 80386 y 80486, los bits MOD especifican también un desplazamiento de 32 bits.
7. Un código binario de 3 bits en los campos REG y R/M especifica los registros cuando MOD = 11. Los registros de 8 bits son: AH, AL, BH, BL, CH, CL, DH y DL; los registros de 16 bits son: AX, BX, CX, DX, SP, BP, DI y SI. Los registros de 32 bits son EAX, EBX, ECX, EDX, ESP, EBP, EDI y ESI.
8. Cuando en el campo R/M describe un modo de memoria, un código de 3 bits selecciona uno de los siguientes modos: [BX + DI], [BX + SI], [BP + DI], [BP + SI], [BX], [BP], [DI] o [SI] para instrucciones de 16 bits en los 8086-80486. En los 80386 y 80486 el campo R/M especifica: EAX, EBX, ECX, EDX, EBP, EDI y ESI o uno de los modos de índice escalado para direccionar los datos en la memoria. Si se selecciona el modo índice escalado (R/M = 100), se añade un byte adicional (byte de índice escalado) a la instrucción original para especificar el registro base, el registro índice y el factor de escala.
9. Todos los modos de direccionamiento de la memoria, en forma implícita direccionan los datos en el segmento de datos salvo que BP direccione la memoria. El registro BP direcciona los datos en el segmento de pila.
10. Los registros de segmento sólo se pueden direccionar con las instrucciones MOV, PUSH o POP. La instrucción MOV puede transferir un registro de segmento a un registro de 16 bits o viceversa. No se permiten las instrucciones MOV CS,reg ni POP CS.
11. Las instrucciones PUSH y POP transfieren datos entre un registro o una localidad de memoria y la pila. Las variantes de estas instrucciones permiten transferir datos inmediatos a la

pila, transferir el registro de banderas a la pila y transferir todos los registros de 16 bits entre la pila y los registros. Cuando se transfieren los datos a la pila, en los 8086-80286 se mueven siempre dos bytes con el byte menos significativo colocado en la localidad SP-2 bytes. Una vez en la pila los datos, SP decrementa en 2. En los 80386 y 80486, también se pueden transferir a la pila 4 bytes de datos de una localidad de la memoria o de un registro.

12. Los códigos de operación que transfieren datos entre la pila y el registro banderas son PUSHF y POPF. Los códigos que transfieren todos los registros de 16 bits entre la pila y los registros son PUSHA y POPA. En los 80386 y 80486, el contenido del registro EFLAGS lo transfieren PUSHFD y POPFD.
13. Las instrucciones LEA, LDS y LES cargan en uno o más registros una dirección efectiva. La instrucción LEA carga una dirección efectiva en cualquier registro de 16 bits; LDS y LES cargan la dirección efectiva en cualquier registro de 16 bits y en DS o ES. En los 80386 y 80486, las instrucciones adicionales incluyen LFS, LGS y LSS que cargan un registro de 16 bits y FS, GS o SS.
14. En las instrucciones para transferencia de cadenas de datos se utilizan uno o ambos DI y SI para direccionar la memoria. La dirección de desplazamiento de DI se encuentra en el segmento extra y la dirección de desplazamiento de SI se encuentra en el segmento de datos.
15. La bandera de dirección (D) selecciona el modo de autoincremento o autodecremento de DI y SI para las instrucciones cadenas. Si se desactiva D con la instrucción CLD, se selecciona el modo de autoincremento; si se activa D con STD, se selecciona el modo de autodecremento. DI y SI, solos o juntos, se incrementan o decrementan en 1 para una operación con byte y en 2 para una operación con palabra.
16. LODS carga a AL, AX o EAX con datos de la localidad de memoria direccionada por SI. STOS carga AL, AX y EAX en la localidad de memoria direccionada por DI; MOVS transfiere un byte o una palabra de la localidad de memoria direccionada por SI a la localidad direccionada por DI.
17. INS transfiere a los datos de un dispositivo de E/S direccionado por DX y a la localidad de memoria direccionada por DI; OUTS transfiere el contenido de la localidad de memoria direccionada por SI y lo envía al dispositivo de E/S direccionado por DX.
18. El prefijo REP se puede añadir a cualquier instrucción en cadena para repetirla. El prefijo REP repite la instrucción en cadena el número de veces que se indica en el registro CX.
19. La instrucción traducir (XLAT) convierte los datos en AL a un número almacenado en la localidad de memoria direccionada por BX más AL.
20. IN y OUT transfieren datos entre AL, AX o EAX y un dispositivo de E/S. La dirección del dispositivo está en la instrucción (puerto fijo) o en el registro DX (puerto variable).
21. El prefijo para cambio de segmento selecciona un registro de segmento distinto para una localidad de memoria, del segmento implícito. Por ejemplo, en la instrucción MOV AX,[BX], se utiliza el segmento de datos, pero en la instrucción MOV AX,ES:[BX] se utiliza el segmento extra debido al prefijo ES. El prefijo para cambio de segmento es la única forma en que se direcciona a los segmentos FS y GS en los microprocesadores 80386 y 80486.
22. La instrucción MOVZX (transferir y extender el cero) y la MOVSX (mover y extender el signo) en los microprocesadores 80386 y 80486 aumentan el tamaño de un byte una palabra o el de una palabra una doble palabra. La versión para extender con ceros aumenta el tamaño del número al colocar ceros a la izquierda. La versión para extender con signo aumenta el tamaño del número porque copia el bit de signo y lo pone en los bits más significativos del número.

23. Los directivos del ensamblador DB (define un byte) DW (define una palabra) DD (define una doble palabra) y DUP (duplicar), almacenan datos en el sistema de la memoria.
24. El directivo EQU (igualar) permite igualar datos o etiquetas con otras etiquetas.
25. El directivo SEGMENT identifica el comienzo del segmento de la memoria y ENDS identifica el final de un segmento.
26. El directivo ASSUME le indica al ensamblador cuáles nombres de segmentos ha asignado el usuario a CS, DS, ES y SS. En los 80386 y 80486 indica también el nombre del segmento para FS y GS.
27. Los directivos PROC Y ENDP indican el comienzo y el final de un procedimiento.
28. Los modelos de memoria se pueden utilizar para acortar un poco el programa, pero son más difíciles de utilizar para programas grandes y para la programación en general. Los modelos de memoria no son compatibles con todos los programas de ensamblador.

3–9 CUESTIONARIO Y PROBLEMAS

1. El primer byte de una instrucción es el ___________ salvo que contenga un prefijo de cambio.
2. Describa la finalidad de los bits D y W que se encuentran en algunas instrucciones en lenguaje de máquina.
3. ¿Qué información especifica el campo MOD en una instrucción en lenguaje de máquina?
4. Si el campo de registro (REG) de una instrucción contiene un 010 y si W = 0, ¿qué registró se selecciona en el supuesto de que la instrucción es en el modo de 16 bits?
5. ¿Cómo se seleccionan los registros de 32 bits en los microprocesadores 80386 y 80486?
6. ¿Qué modo de direccionamiento de la memoria se especifica con R/M = 001 y con MOD = 00 para una instrucción de 16 bits?
7. Determine el registro de segmento implícito asignado a:
   (a) SP
   (b) BX
   (c) DI
   (d) BP
   (e) SI
8. Convierta un 8B07H de lenguaje de máquina, a lenguaje ensamblador.
9. Convierta un 8B1E004CH de lenguaje de máquina a lenguaje ensamblador.
10. Si aparece una instrucción MOV SL,[BX + 2], ¿cuál es su equivalente en lenguaje de máquina?
11. Si aparece una instrucción MOV ESI,[EAX] en un programa para los microprocesadores 80386 y 80486, que trabaja con un modo de instrucción de 16 bits, ¿cuál es su equivalente en lenguaje de máquina?
12. ¿Qué hay de incorrecto en una instrucción MOV CS, AX?
13. PUSH y POP siempre transfieren un número de _____bits entre la pila y un registro o una localidad de memoria en el microprocesador 8086-80286.
14. ¿Qué registro de segmento no se puede extraer de la pila?
15. ¿Qué registros se mueven dentro de la pila con una instrucción PUSHA?
16. ¿Qué registros se mueven dentro de la pila con una instrucción PUSHAD?

17. Describa el funcionamiento de cada una de las siguientes instrucciones:
    (a) PUSH AX
    (b) POP ESI
    (c) PUSH [BX]
    (d) PUSHFD
    (e) POP DS
    (f) PUSH 4
18. Explique lo que sucede cuando se ejecuta la instrucción PUSH BX. Cerciórese de mostrar en dónde se almacenan BH y BL. (Suponga que SP = 0100H y SS = 0200H.)
19. Repita la pregunta 18 para la instrucción PUSH EAX.
20. La instrucción POP de 16 bits (excepto POPA) incrementa SP en __________.
21. ¿Qué valores aparecen en SP y SS si el apuntador de la pila direcciona la localidad de memoria 02200H?
22. Compare el funcionamiento de una instrucción MOV DI,NUMB con el de una instrucción LEA DI,NUMB.
23. ¿Cuál es la diferencia entre una instrucción LEA SI,NUMB y una instrucción MOV SI,OFFSET NUMB?
24. ¿Cuál es más eficiente: MOV con un OFFSET o LEA?
25. Describa cómo funciona la instrucción LDS BX,NUMB.
26. ¿Cuál es la diferencia entre las instrucciones LDS y LSS?
27. Escriba una secuencia de instrucciones que transfiera el contenido de las localidades NUMB y NUMB+1 del segmento de datos, a BX, DX y SI.
28. ¿Cuál es la finalidad de la bandera de dirección?
29. ¿Qué instrucciones activan y desactivan la bandera de dirección?
30. Las instrucciones para cadenas utilizan DI y SI para direccionar datos de la memoria, ¿en cuál segmento de memoria?
31. Explique el funcionamiento de la instrucción LODSB.
32. Explique el funcionamiento de la instrucción STOSW.
33. Explique el funcionamiento de la instrucción OUTSB.
34. ¿Qué logra el prefijo REP y con qué tipo de instrucción se utiliza?
35. Escriba una secuencia de instrucciones que transfieran 12 bytes de datos de una zona de la memoria direccionada por FUERZA, a una zona de memoria direccionada por DESTINO.
36. ¿Dónde se almacena la dirección de E/S (numero de puerto) para una instrucción INSB?
37. Seleccione una instrucción en lenguaje ensamblador que intercambie el contenido del registro EBX con el del registro ESI.
38. ¿Aparecerán, por lo general, las instrucciones LAHF y SAHF en muchos programas?
39. Explique la forma en que la instrucción XLAT transforma el contenido del registro AL.
40. Escriba un programa corto en que emplee la instrucción XLAT para convertir los números 0 a 9 en BCD a los 30H a 39H codificados en ASCII. Almacene los datos codificados en ASCII en una TABLA.
41. Explique qué hace la instrucción IN AL,12H.
42. Explique cómo funciona la instrucción OUT DX,AX.
43. ¿Qué es un prefijo para cambio de segmento?
44. Seleccione una instrucción que transfiera un byte de datos de la localidad de memoria direccionada con el registro BX en el segmento extra, al registro AH.

45. Escriba una secuencia de instrucciones que intercambie el contenido de AX con BX, ECX y EDX y SI con DI.
46. ¿Qué es un directivo en lenguaje ensamblador?
47. Describa la finalidad de los siguientes directivos en lenguaje ensamblador: DB, DW y DD.
48. Seleccione un directivo en lenguaje ensamblador que reserve 30 bytes de memoria para el arreglo LIST1.
49. Describa la finalidad del directivo EQU.
50. ¿Cuál es la finalidad del directivo .386?
51. ¿Cuál es la finalidad del directivo .MODEL?
52. Si el comienzo de un segmento se señala con DATA, ¿qué tipo de organización de memoria está en vigor?
53. Si el directivo SEGMENT señala el comienzo de un segmento, ¿qué tipo de organización de memoria está en vigor?
54. ¿Qué hace la INT 21H si AH contiene un 4CH?
55. ¿Qué directivos indican el comienzo y el final de un procedimiento?
56. Explique la finalidad del enunciado USES cuando se aplica a un procedimiento con la versión 6.0 del MASM.
57. Establezca un procedimiento cercano que almacene AL en cuatro localidades consecutivas en la memoria, dentro del segmento de datos direccionado por el registro DI.
58. Desarrolle un procedimiento lejano que copie datos de tamaño palabra en la localidad de memoria CS:DATA1 y los transfiera en AX, BX, CX, DX y SI.

CAPITULO 4

Instrucciones aritméticas y lógicas

INTRODUCCION

En este capítulo se examinan las instrucciones aritméticas y lógicas que se encuentran en el conjunto de instrucciones para los procesadores 8086-80486. Las instrucciones aritméticas incluyen suma, resta, multiplicación, división, comparación, negación, incrementar y decrementar. Las instrucciones lógicas incluyen: AND, OR, OR exclusivo, NOT, corrimientos, rotaciones y la comparación lógica (TEST). Además se presentan las instrucciones para el 80386/80486, SHRD, SHLD, pruebas de bits y rastreo de bits.

También se presentan instrucciones para comparación de cadenas, que se utilizan para rastrear datos contenidos en tablas y para comparar secciones de memoria. Ambas tareas se efectúan con eficiencia con las instrucciones para rastrear y comparar en cadenas.

Si el lector ya conoce algún microprocesador de 8 bits, reconocerá que el conjunto de instrucciones para los 8086-80486 son mucho mejores. Aunque se trate del primer microprocesador con que trabaje, aprenderá con rapidez que el microprocesador tiene un conjunto de instrucciones aritméticas y lógicas potentes y fáciles de usar.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo el lector podrá:

1. Utilizar las instrucciones aritméticas y lógicas para operaciones binarias, en BCD y ASCII.
2. Utilizar AND, OR Y OR-exclusivo para manipulación de bits.
3. Utilizar las instrucciones para corrimiento y rotación.
4. Explicar el funcionamiento de instrucciones de doble precisión, para corrimiento prueba y rastreo de bits.
5. Comprobar el contenido de una tabla para buscar un elemento con las instrucciones para cadena.

4–1 SUMA, RESTA Y COMPARACION

La mayor parte de las instrucciones aritméticas que se encuentran en cualquier microprocesador incluyen suma, resta y comparación. Los microprocesadores 8086-80486 no son diferentes. En esta sección se describen y definen estas instrucciones. También se muestra su empleo en la manipulación de datos en registros y memoria.

Suma

La suma tiene muchas formas en los microprocesadores 8086-80486. En esta sección se detalla el empleo de la instrucción ADD para suma binaria de 8, 16 y 32 bits. Se presenta otra forma de la suma, llamada *suma con acarreo* junto con la instrucción ADC; ésta permite efectuar sumas de casi cualquier ancho. También se presenta Incrementar (INC) que es un tipo especial de suma. En la sección 4-3 se examinan otras formas para sumar tales como BCD y ASCII.

En la tabla 4-1 se ilustran los modos de direccionamiento disponibles en la instrucción ADD. (Los modos de direccionamiento incluyen a casi todos los mencionados en el capítulo 2.) Dado que hay más de 1000 variantes de la instrucción ADD en el conjunto de instrucciones, es imposible incluirlas todas en esta tabla. Los únicos tipos de sumas que no se permiten son las de memoria a memoria y registros de segmento. Los registros de segmento sólo se pueden transferir, salvar o recuperar. Se debe tener en cuenta que, igual que en todas las demás instrucciones, los registros de 32 bits sólo están disponibles para los microprocesadores 80386 y 80486.

Suma de registros. En el ejemplo 4-1 se presenta un programa sencillo en el cual se emplea suma de registros para sumar cierto número de registros entre sí. En este ejemplo, se suma el contenido de AX, BX, CX y DX para formar una suma de 16 bits que se almacena en el registro AX.

EJEMPLO 4–1

```
0000 03 C3        ADD   AX,BX
0002 03 C1        ADD   AX,CX
0004 03 C2        ADD   AX,DX
```

Siempre que se ejecutan la mayor parte de las instrucciones aritméticas y lógicas, el contenido del registro de banderas. Las banderas muestran el resultado de la operación aritmética. Cualquier instrucción ADD modifica el contenido de las banderas de signo, cero, acarreo, acarreo auxiliar, paridad y sobreflujo. (Se debe tener en cuenta que los bits de bandera no cambian en la mayor parte de las instrucciones para transferencia de datos presentadas en el capítulo 3.)

Suma inmediata. La suma inmediata se utiliza siempre que se suman datos constantes o conocidos. En el ejemplo 4-2 aparece una suma inmediata de 8 bits. En este ejemplo, primero se carga DL con 12H con el empleo de una instrucción para transferencia inmediata. Luego se suma un 33H al 12H en DL, con una suma inmediata. Después de la suma, el total (45H) se transfiere al registro DL y las banderas cambian como sigue:

Z = 0 (resultado no es cero)

C = 0 (no hay acarreo)

TABLA 4–1 Instrucciones para sumar

Instrucción	*Comentario*
ADD AL,BL	AL = AL + BL
ADD CX,DI	CX = CX + DI
ADD EBP,EAX	EBP = EBP + EAX
ADD CL,44H	CL = CL + 44H
ADD BX,35AFH	BX = BX + 35AFH
ADD EDX,12345H	EDX = EB + X00012345H
ADD [BX],AL	Se suma AL al contenido de la localidad con desplazamiento en el segmento de datos direccionado por BX y el resultado se almacena en la misma localidad de la memoria
ADD CL,[BP]	El contenido de la localidad con desplazamiento en el segmento de pila direccionada por BP se suma a CL y el resultado se almacena en CL
ADD AL,[EBX]	El contenido de la localidad con desplazamiento en el segmento de datos direccionada por EBX, se suma a AL y el resultado se almacena en AL
ADD BX,[SI + 2]	El contenido de tamaño-palabra de la localidad en el segmento de datos direccionada por SI más 2 se suma a BX y el resultado se almacena en la misma localidad de la memoria
ADD CL,TEMP	El contenido de la localidad TEMP en el segmento de datos se suma a CL y el resultado se almacena en CL
ADD BX,TEMP[DI]	El contenido de tamaño-palabra de la localidad en el segmento de datos direccionada por TEMP más DI se suma a BX y el resultado se almacena en la misma localidad en la memoria
ADD [BX + DI],DL	El byte de memoria del segmento de datos direccionado por BX + DI es la suma de ese byte más DL
ADD BYTE PTR [DI],3	Agrega 3 al contenido de la localidad de la memoria de tamaño byte, direccionada por DI dentro del segmento de datos
ADD BX,[EAX+2*ECX]	La dirección de la palabra del segmento de memoria de datos por la suma de 2 multiplicado por ECX y EAX se suma a BX

A = 0 (no hay semiacarreo)
S = 0 (resultado es positivo)
P = 0 (paridad impar)
O = 0 (no hay sobreflujo)

EJEMPLO 4–2

```
0006 B2 12        MOV   DL,12H
0008 80 C2 33     ADD   DL,33H
```

Suma de memoria a registro. Supóngase que una aplicación requiere sumar datos de memoria al registro AL. En el ejemplo 4-3 se muestra cómo se suman 2 bytes consecutivos de datos de la memoria, almacenados en las localidades con desplazamientos NUM y NUM+1 en el registro AL.

EJEMPLO 4–3

```
0000 BF 0000 R          MOV  DI,OFFSET NUMB     ;direccionar NUM
0003 B0 00              MOV  AL,0               ;borrar suma
0005 02 05              ADD  AL,[DI]            ;sumar NUM
0007 02 45 01           ADD  AL,[DI+1]          ;sumar NUM+1
```

La secuencia de instrucciones primero carga el contenido del registro índice destino (DI) con el desplazamiento de la dirección NUM. El registro DI, utilizado en este ejemplo, direcciona los datos en el segmento de datos a partir de la localidad NUM en la memoria. Luego, la instrucción ADD AL, [DI] suma el contenido de la localidad NUM de la memoria a AL. Esto ocurre porque DI direcciona a la localidad NUM de la memoria y la instrucción suma su contenido en AL. Asimismo la instrucción ADD AL,[DI+1] suma el contenido de la localidad NUM+1 de la memoria, al registro AL. Después de que se ejecutan ambas instrucciones ADD, el resultado aparece en el registro AL como la suma de NUM más NUM+1.

Suma de arreglos. Los arreglos en memoria son listas de datos que hay en la memoria. Supóngase que un arreglo (ARREY) de datos de 10 bytes numerados del elemento 0 al elemento 9. En el ejemplo 4-4 se muestra un programa que suma el contenido de los elementos 3, 5 y 7 de la matriz. (El programa y los elementos de arreglo que se suman se han escogido para mostrar el empleo de algunos de los modos de direccionamiento del microprocesador.)

EJEMPLO 4–4

```
0000 B0 00              MOV  AL,0               ;borrar suma
0002 BE 0003            MOV  SI,3               ;direccionar elemento 3
0005 02 84 0002 R       ADD  AL,ARRAY[SI]       ;sumar elemento 3
0009 02 84 0004 R       ADD  AL,ARRAY[SI+2]     ;sumar elemento 5
000D 02 84 0006 R       ADD  AL,ARRAY[SI+4]     ;sumar elemento 7
```

Con este ejemplo, primero se «borra» AL para que quede en cero y se pueda emplear para acumular la suma. Después se carga un 3 en el registro SI para direccionar inicialmente el elemento 3 del arreglo. La instrucción ADD AL,ARREY[SI] suma el contenido del elemento 3 del arreglo y lo agrega a la suma que hay en AL. Las instrucciones que siguen suman los elementos 3 y 7 del arreglo a la suma en AL, con el empleo de un 3 en SI más un desplazamiento de 2 para direccionar al elemento 5 y un desplazamiento de 4 para direccionar al elemento 7.

Supóngase que un arreglo de datos contiene números de 16 bits para formar una suma de 16 bits en el registro AX. En el ejemplo 4-5 se muestra una secuencia corta de instrucciones, escritas para los microprocesadores 80386 y 80486, con el empleo de direccionamiento de índice escalado para sumar los elementos 3, 5 y 7 de una zona de la memoria llamada ARREY (arreglo). En este ejemplo se carga EBX con la dirección ARREY y ECX contiene el número del elemento del arreglo.

EJEMPLO 4–5

```
0000 66| BB 00000000 R   MOV  EBX,OFFSET ARREGLO  ;direcciona ARREGLO (Matriz)
0006 66| B9 00000003     MOV  ECX,3               ;direcciona elemento 3
000C 67& 8B 04 4B        MOV  AX,[EBX+2*ECX]      ;obtener elemento 3
0010 66| B9 00000005     MOV  ECX,5               ;direcciona elemento 5
0016 67& 03 04 4B        ADD  AX,[EBX+2*ECX]      ;sumar elemento 5
001A 66| B9 00000007     MOV  ECX,7               ;direcciona elemento 7
0020 67& 03 04 4B        ADD  AX,[EBX+2*ECX]      ;sumar elemento 7
```

TABLA 4-2 Instrucciones para incremento

Instrucción	*Comentario*
INC BL	BL = BL + 1
INC SP	SP = SP + 1
INC EAX	EAX = EAX + 1
INC BYTE PTR [BX]	Se incrementa el contenido byte de la localidad de la memoria direccionada por BX en el segmento de datos
INC WORD PTR [SI]	Se incrementa el contenido palabra de la localidad de memoria direcccionada por SI, en el segmento de datos
INC DWORD PTR [ECX]	Se incrementa el contenido doble palabra de la localidad en memoria del segmento de datos direccionada por ECX
INC DATO1	Se incrementa el contenido de DATO1

Suma de incremento. La suma de incremento (INC) agrega un 1 a un registro o a una localidad de la memoria. La instrucción INC puede sumar un 1 a cualquier registro o localidad de la memoria, excepto a un registro de segmento. En la tabla 4-2 se muestran algunas de las posibles formas de la instrucción de incremento disponibles en los microprocesadores 8086 a 80486. Al igual que las otras instrucciones descritas, es imposible mostrar todas las variantes de la instrucción INC debido a su gran cantidad.

Con incrementos indirectos en la memoria, el tamaño de los datos se debe describir con el empleo de los directivos BYTE PTR, WORD PTR o DWORD PTR. La razón es que el programa ensamblador no puede determinar si, por ejemplo, la instrucción INC [DI] es un incremento de tamaño de byte, palabra o doble palabra. La instrucción INC BYTE PTR[DI] indica con claridad datos de memoria de tamaño byte; la instrucción INC WORD PTR[DI] indica sin lugar a duda un dato de memoria de tamaño palabra; la instrucción INC DWORD PTR[DI] incrementa datos de tamaño doble palabra.

EJEMPLO 4-6

```
0000 BF 0000 R        MOV  DI,OFFSET NUMB     ;direccionar NUM
0003 B0 00            MOV  AL,0               ;borrar la suma
0005 02 05            ADD  AL,[DI]            ;sumar NUM
0007 47               INC  DI                 ;direccionar NUM+1
0008 02 05            ADD  AL,[DI]            ;direccionar NUM+1
```

En el ejemplo 4-6 se muestra la modificación del programa del ejemplo 4-3 para utilizar la instrucción de incrementar para direccionar a NUM y NUM+1. En este caso, una instrucción INC DI cambia el contenido del registro DI, de la dirección de desplazamiento NUM a la dirección de desplazamiento NUM+1. La diferencia entre estos programas es la forma en que se determina la dirección de estos datos por medio del contenido del registro DI y con el empleo de la instrucción para incrementar.

Las instrucciones para incrementar influyen en los bits de bandera igual que la mayor parte de las otras operaciones aritméticas y lógicas. La diferencia es que las instrucciones para incrementar no influyen en el bit de la bandera de acarreo. El acarreo no se altera porque la instrucción

TABLA 4–3 Instrucciones para suma con acarreo

Instrucción	*Comentario*
ADC AL,AH	AL = AL+ AH + carry
ADC CX,BX	CX = CX + BX + carry
ADC EBX,EDX	EBX = EBX + EDX + carry
ADC [BX], DH	El contenido bytes de la localidad de la memoria en el segmento de datos direccionada por BX se suma con DH, y el acarreo. El resultado se almacena en la memoria
ADC BX,[BP + 2]	Se suman BX y el contenido palabra de la localidad de memoria en el segmento de pila direccionada por BP y el resultado se almacena en BX
ADC ECX,[EBX]	Se suman con acarreo ECX y el contenido doble palabra de la localidad de la memoria, en el segmento de datos direccionado por EBX y el resultado se almacena en ECX

Nota: Los registros y modos de direcccionamiento de 32 bits sólo se utilizan en los 80386 y 80486.

para incrementar se utiliza a menudo en programas que dependen del contenido de la bandera de acarreo.

Suma con acarreo. Una instrucción de suma, con acarreo (ADC), suma el bit de la bandera (C) de acarreo a los datos del operando. Esta instrucción casi siempre aparece en programas que suman elementos de un ancho mayor de 16 bits en los 8086-80286 o de 32 bits en los 80386 y 80486.

En la tabla 4-3 se presentan algunas instrucciones de suma con acarreo con comentarios de su funcionamiento. Al igual que la instrucción ADD, la ADC afecta las banderas después de la suma.

Supóngase que se escribe un programa para 8086-80286 que suma el número de 32 bits en BX y AX, al número de 32 bits en DX y CX. En la figura 4-1 se ilustra esta suma a fin de poder entender la colocación y la función de la bandera de acarreo. Esta suma no se puede efectuar sin sumar el bit de la bandera de acarreo, porque los 8086-80286 sólo suman números de 8 o de 16 bits. En el ejemplo 4-7 se muestra la forma en que ocurre esa suma con un programa. En este caso, el contenido de los registros AX y CX se suma para formar los 16 bits menos significativos de la suma total. Esta suma puede o no generar un acarreo, el cual aparece en la bandera de acarreo, si el total de la suma es mayor de FFFFH. Debido a que es imposible predecir un acarreo, los 16 bits más significativos de esta suma se suman con la bandera de acarreo por medio de la instrucción ADC; esta suma el uno o el cero de la bandera de acarreo con los 16 bits más significativos del resultado. Con este programa se suman BX y AX con DX y CX y la suma aparece en BX-AX.

EJEMPLO 4–7

```
0000 03 C1          ADD  AX,CX
0002 13 DA          ADC  BX,DX
```

Supóngase que se vuelve a escribir el mismo programa para los microprocesadores 80386 y 80486, pero modificado para sumar dos números de 64 bits. Los cambios requeridos son el empleo de los registros extendidos para contener los datos así como para efectuar las modificaciones

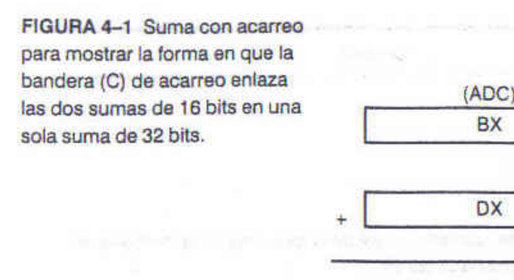
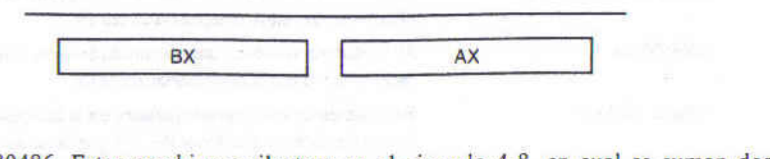



FIGURA 4–1 Suma con acarreo para mostrar la forma en que la bandera (C) de acarreo enlaza las dos sumas de 16 bits en una sola suma de 32 bits.

para los 80386 y 80486. Estos cambios se ilustran en el ejemplo 4-8, en cual se suman dos números de 64 bits.

EJEMPLO 4–8

```
0000 66| 03 C1             ADD   EAX,ECX
0003 66| 13 DA             ADC   EBX,EDX
```

Intercambiar y sumar para el microprocesador 80486. En el conjunto de instrucciones para el 80486 aparece un nuevo tipo de suma llamada intercambiar y sumar (XADD), la cual suma datos de la fuente a los del destino y almacena la suma en el destino igual que con cualquier otra suma. La diferencia es que, después de que ocurre la suma, se copia el valor original del destino y se pone en el operando fuente. Esta instrucción es una de las pocas que cambian la fuente.

Por ejemplo, si BL = 12H y DL = 02H y se ejecuta la instrucción XADD BL, DL, el registro BL contiene la suma de 14H y DL se convierte en 12H. Se genera la suma de 14H y el destino original de 12H sustituye a la fuente. Esta instrucción funciona con cualquier tamaño de registro y cualquier operando de memoria, igual que la instrucción ADD normal.

Resta

En el conjunto de instrucciones de los 8086-80286 aparecen muchas formas de resta (SUB). Estas formas utilizan cualquier modo de direccionamiento con datos de 8, 16 o 32 bits. Una forma especial de la resta (decremento) resta o decrementa un 1 en cualquier registro o localidad de la memoria. En la sección 4-5 se ilustra la forma en que se restan los datos en BCD y en ASCII. Igual que con la suma, en ocasiones hay que restar números de un ancho mayor de 16 o de 32 bits. La instrucción de resta con préstamo (SBB) efectúa este tipo de resta.

En la tabla 4-4 se presentan muchos modos de direccionamiento permitidos con la instrucción (SUB) para restar. Hay más de 1000 posibles instrucciones para la resta, demasiadas para una lista. Casi los únicos tipos de resta no permitidos son de memoria a memoria y registro de segmento. Igual que otras instrucciones aritméticas, la instrucción para resta influye en los bits de bandera.

Resta de registros. En el ejemplo 4-9 se muestra un programa que efectúa restas en registros; en este ejemplo se muestra la resta de contenido de 16 bits de los registros CX y DX, del contenido

TABLA 4–4 Instrucciones para resta

Instrucción	*Comentario*
SUB CL,BL	CL = CL - BL
SUB AX,SP	AX = AX - SP
SUB ECX,EBP	ECX = ECX - EBP
SUB DH,6FH	DH = DH - 6FH
SUB AX,0CCCCH	AX = AX - 0CCCCH
SUB EAX,23456H	EAX = EAX - 00023456H
SUB [DI],CH	Se resta CH del contenido byte de la localidad de memoria en el segmento de datos direccionada por DI
SUB CH,[BP]	Se resta de CH el contenido de bytes de la localidad de memoria en el segmento de pila direccionada por BP
SUB AH,TEMP	Se resta de AH el contenido byte de la localidad TEMP de la memoria en el segmento de datos
SUB DI,TEMP[BX]	Se resta de DI los datos de la localidad de memoria, en el segmento de datos, direccionada por TEMP más BX
SUB ECX,DATO1	Se resta de ECX el contenido doble palabra de la localidad de memoria en el segmento de datos direccionada por DATO1

Nota: Los registros y modos de direccionamiento de 32 bits sólo se utilizan en los 80386 y 80486.

del registro BX. Después de cada resta, el microprocesador modifica el contenido del registro de banderas. Las banderas cambian para la mayor parte de las operaciones aritméticas y lógicas.

EJEMPLO 4–9

```
0000 2B D9          SUB  BX,CX
0002 2B DA          SUB  BX,DX
```

Resta inmediata. Igual que para las sumas, el microprocesador permite que haya operandos inmediatos para la resta de datos constantes. En el ejemplo 4-10 se presenta un programa corto para restar 44H de 22H. En este caso primero se carga el 22H en CH, con el empleo de una instrucción para transferir datos inmediatos. A continuación la instrucción SUB, empleando el dato 44H inmediato, resta 44H de 22H. Después de la resta, la diferencia (DEH) se transfiere al registro CH. Las banderas cambian como sigue para esta resta:

Z = 0 (el resultado no es cero)

C = 1 (préstamo)

A = 1 (semipréstamo)

S = 1 (resultado negativo)

P = 1 (paridad par)

O = 0 (no hay sobreflujo)

EJEMPLO 4–10

```
0000 B5 22          MOV  CH,22H
0002 80 ED 44       SUB  CH,44H
```

Ambas banderas de acarreo (C y A) contienen préstamos en lugar de acarreos como después de una suma. Se verá que en este ejemplo no hay sobreflujo. Con este ejemplo se restó un 44H (+68) de un 22H (+34) y el resultado fue un DEH (–34). Debido a que el resultado correcto de 8 bits con signo es –34, no hay sobreflujo en este ejemplo. Un sobreflujo de 8 bits sólo ocurre si el resultado con signo es mayor de +127 o menor de –128.

Decremento. El decremento (DEC) resta un 1 de un registro o del contenido de una localidad de memoria. En la tabla 4-5 se presentan algunas instrucciones de decremento que la ilustran para registros y memoria.

Las instrucciones para decrementar datos indirectos en memoria, requieren BYTE PTR, WORD PTR o DWORD PTR, porque el ensamblador no puede distinguir un byte de una palabra, cuando un registro índice direcciona la memoria. Por ejemplo DEC[SI] es vaga, porque el ensamblador no puede determinar si la localidad direccionada por SI es un byte o una palabra. El tamaño de los datos se revela con el empleo de DEC BYTE PTR[SI], DEC WORD PTR [DI] o DEC DWORD PTR [SI].

Resta con préstamo. Una instrucción de resta con préstamo (SBB) funciona igual que una resta normal, excepto que la bandera (C) de acarreo también se resta de la diferencia. El empleo más común de esta instrucción es para restas con un ancho mayor de 16 bits en los 8086-80286 o de un ancho mayor de 32 bits en los 80386 y 80486. Las restas «anchas» requieren que los préstamos se propaguen en la resta, igual que en las sumas «anchas» se propagó el acarreo.

TABLA 4–5 Instrucciones para decrementar

Instrucción	*Comentario*
DEC BH	BH = BH – 1
DEC SP	SP = SP – 1
DEC ECX	ECX = ECX – 1
DEC BYTE PTR [DI]	Se decrementa el contenido byte de la localidad de memoria en el segmento de datos direccionada por DI
DEC WORD PTR [BP]	Se decrementa el contenido palabra de la localidad de memoria en el segmento de datos direccionada por BP
DEC DWORD PTR [EBX]	Se decrementa el contenido dobles palabras de la localidad de memoria en el segmento de datos direccionada por EBX
DEC NUM	Se decrementa el contenido de la localidad NUM en la memoria en el segmento de datos. La forma en que se defina NUM determina si es un decremento de byte o de palabra

Nota: Los registros y modos de direccionamiento de 32 bits sólo se utilizan en los 80386 y 80486.

TABLA 4–6 Instrucciones para restar con préstamo

Instrucción	*Comentario*
SBB AH,AL	AH = AH – AL – carry
SBB AX,BX	AX = AX – BX – carry
SBB EAX,EBX	EAX = EAX – EBX – carry
SBB CL,3	CL = CL – 3 – carry
SBB BYTE PTR [DI],3	3 y el acarreo se restan del contenido byte de la localidad en la memoria del segmento de datos direccionada por DI
SBB[DI],AL	AL y acarreo se restan del contenido byte de la localidad de memoria en el segmento de datos direccionada por DI
SBB DI,[BP+ 2]	Se resta de DI el contenido palabra de la localidad de memoria en el segmento de pila, direccionada por BP + 2, y el acarreo
SBB AL,[EBX+ECX]	Se resta de AL el contenido byte de la localidad en la memoria del segmento de datos direccionada por EBX y ECX

Nota: Los registros y modos de direccionamiento de 32 bits sólo se utilizan en los 80386 y 80486.

En la tabla 4-6 se presentan muchas instrucciones de SBB y los comentarios de su funcionamiento. SBB, al igual que SUB, afecta las banderas. Se debe tener en cuenta que la instrucción de resta de memoria inmediata en esta tabla, requiere emplear BYTE PTR, WORD PTR o DWORD PTR.

Cuando el número de 32 bits contenido en BX y AX se resta del número de 32 bits que hay en DI y SI, la bandera de acarreo propaga el préstamo entre las dos restas de 16 bits requeridas para efectuar la operación en los microprocesadores 8086-80486. En la figura 4-2 se ilustra la forma en que el préstamo se propaga con la bandera (C) de acarreo para esta tarea.

El ejemplo 4-11 muestra cómo se efectúa esta resta con un programa. Con esta resta «ancha», los datos menos significativos de 16 o de 32 bits se restan con la instrucción SUB. Todos los datos más significativos subsecuentes se restan con la instrucción SBB. En el ejemplo se utiliza la instrucción SUB para restar DI de AX y, luego, SBB para restar con préstamo a SI de BX.

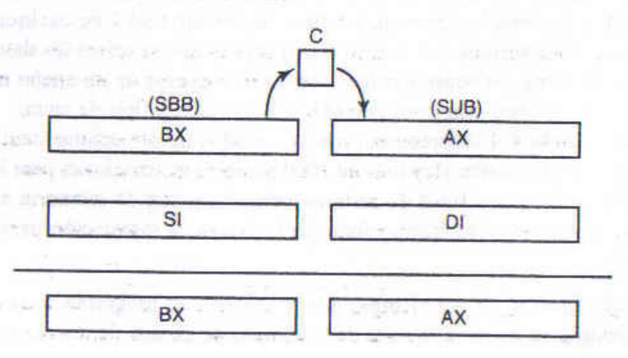


FIGURA 4–2 Resta con préstamo que muestra la forma en que la bandera (C) de acarreo propaga el préstamo.

EJEMPLO 4–11

```
0004 2B C7            SUB  AX,DI
0006 1B DE            SBB  BX,SI
```

Comparación

La instrucción (CMP) para comparación es una resta que sólo afecta los bits de bandera. La comparación es útil para comprar el contenido de un registro o una localidad de memoria contra otro valor. La CMP suele ir seguida por una instrucción de brinco (JMP) condicional que prueba las condición de los bits de bandera.

En la tabla 4-7 se presentan diversas instrucciones para comparación en las que se emplean los mismos modos de direccionamiento que para las instrucciones de suma y resta ya descritas. También en este caso, las únicas formas de CMP no permitidas son la comparación de memoria a memoria y de registros de segmento.

EJEMPLO 4–12

```
0000 3C 10            CMP  AL,10H   ;comparar con 10H
0002 73 1C            JAE  SUBER    ;si es 10H o mayor
```

TABLA 4–7 Instrucciones para comparación

Instrucción	*Comentario*
CMP CL,BL	Resta BL de CL; no cambian BL ni CL, pero sí cambian las banderas
CMP AX,SP	Resta SP de AX; no cambian AX ni SP, pero sí cambian las banderas
CMP EBP,ESI	Resta ESI de ESP; no cambian ESP ni ESI, pero sí cambian las banderas
CMP AX,OCCCCH	Resta 0CCCCH de AX; no cambia AX, pero sí cambian las banderas
CMP [DI],CH	Resta CH del contenido de bytes de la localidad de memoria del segmento de datos direccionada por DI; no cambian CH ni la memoria, pero sí cambian las banderas
CMP CL,[BP]	Resta de CL, el contenido byte de la localidad de memoria en el segmento de pila direccionada por BP; no cambian CL ni la memoria, pero sí cambian las banderas
CMP AH,TEMP	Resta de AH, el contenido byte de la localidad TEMP de memoria en el segmento de datos; no cambian AH ni la memoria, pero sí cambian las banderas
CMP DI,TEMP [BX]	Resta de DI, el contenido palabra de la localidad de la memoria en el segmento de datos direccionada por TEMP + BX; no cambian DI ni la memoria, pero sí cambian las banderas
CMP AL,[EDI+ESI]	Resta de la localidad de memoria en el segmento de datos direccionada por EDI y ESI; el contenido byte en la memoria del segmento de datos; no cambian AL ni la memoria, pero sí cambian las banderas

Nota: Los registros y modos de direccionamiento de 32 bits sólo se utilizan en los 80386 y 80486.

En el ejemplo 4-12 se presenta una comparación seguida por una instrucción condicional para brinco. En este ejemplo, el contenido de AL se compara con 10H. Las instrucciones condicionales para brinco que a menudo van después de la de comparación son: JA (brinca si está por arriba) o JB (brinca si está por abajo). Si JA va después de la instrucción CMP, ocurre el brinco si el valor en AL está por arriba de 10H. Si JB va después de la instrucción CMP, ocurre el salto si el valor en AL está por abajo de 10H. En este ejemplo, la instrucción JAE va después de la de comparación. Esta instrucción hace que el programa continúe en la localidad SUBER de la memoria, si el valor en AL es 10H o mayor. También hay una instrucción JBE (saltar si es menor o igual) que podría seguir a la de comparación para brincar, si el resultado es menor o igual a 10H. En el capítulo 5 se dan mayores detalles de la instrucción para comparación y de las instrucciones para brincos condicionales.

Comparar e intercambiar. La instrucción de comparar e intercambiar (CMPXCHG) que sólo se encuentra en las instrucciones para el 80486, compara el operando destino con el acumulador. Si son iguales, se copia el operando fuente en el destino. Si no son iguales, se copia el operando destino y se pone en el acumulador. Esta instrucción funciona con datos de 8, 16 o 32 bits.

La instrucción CMPXCHG CX,DX es un ejemplo de esa instrucción. Esta instrucción primero compara los contenidos de CX y de AX. Si CX es igual a AX, DX se copia en AX. Si CX no es igual a AX se copia en AX. Esta instrucción X también compara AL con datos de 8 bits y a EAX con datos de 32 bits si el operando es de 8 o de 32 bits.

4–2 MULTIPLICACION Y DIVISION

Sólo los microprocesadores más modernos como el 80486, 80386 y 80286, y los primeros 8086-8088, contienen instrucciones para multiplicación y división. En los primeros microprocesadores de 8 bits no se podían hacer multiplicaciones o divisiones directas. Se necesitaba un programa para multiplicar o dividir con el empleo de una serie de corrimientos y de sumas o restas. Debido a que los fabricantes de microprocesadores se percataron de esta deficiencia, incluyeron instrucciones para multiplicación y división en los conjuntos de intrusiones de sus nuevos microprocesadores.

Multiplicación

La multiplicación se efectúa en bytes, palabras o dobles palabras y pueden ser enteros con signo (IMUL) o sin signo (MUL). Se debe tener en cuenta que sólo en los 80386 y 80486 se pueden multiplicar dobles palabras de 32 bits. El producto después de una multiplicación es siempre de doble ancho. Si se multiplican dos números de 8 bits, se genera un producto de 16 bits; si se multiplican dos números de 16 bits, se genera un producto de 32 bits y si se multiplican dos números de 32 bits, se genera un producto de 64 bits.

Algunos bits (O y C) de bandera cambian cuando se ejecuta la instrucción de multiplicar y producen resultados predecibles. Las otras banderas también cambian, pero sus resultados son impredecibles y, por tanto, no se usan. En una multiplicación de 8 bits, si los 8 bits más significativos del resultado son 0, los bits de bandera O y C son iguales a 0. Estos bits de bandera muestran que el resultado tiene un ancho de 8 bits o de 16 bits. En una multiplicación de 16 bits,

si los 16 bits más significativos del producto son 0, entonces C y O son 0. En una multiplicación de 32 bits, C y O indican que los 32 bits más significativos del producto son cero.

Multiplicación de 8 bits. En la multiplicación de 8 bits, con o sin signo, el multiplicando está siempre en el registro AL. El multiplicador puede ser cualquier registro de 8 bits o cualquier localidad de la memoria. No se permite la multiplicación inmediata, salvo que la instrucción para multiplicación inmediata con signo especial, que se describe más adelante en esta sección, aparezca en un programa. La instrucción para multiplicación contiene un operando porque siempre multiplica operando veces el contenido del registro AL. Un ejemplo es la instrucción MUL BL que multiplica el contenido sin signo de AL por el contenido sin signo de BL. Después de la multiplicación, el producto sin signo se coloca en AX, un producto de doble ancho. En la tabla 4-8 aparecen algunas instrucciones para multiplicación de 8 bits.

Supóngase que BL y CL contienen, cada uno, números de 8 bits sin signo, que se deben multiplicar para formar un producto de 16 bits almacenado en DX. Este procedimiento no se puede lograr con una sola instrucción, porque sólo se puede multiplicar un número por el contenido del registro AL para una multiplicación de 8 bits. En el ejemplo 4-13 se presenta un programa breve que genera DX = BL × CL. En este ejemplo se cargan los registros BL y CL con los datos 5 y 10 del ejemplo. El producto, un 50, se transfiere de AX a DX después de la multiplicación, por medio de la instrucción MOV DX,AX.

EJEMPLO 4–13

```
0000 B3 05          MOV  BL,5    ;cargar datos
0002 B1 0A          MOV  CL,10
0004 8A C1          MOV  AL,CL   ;posicionar los datos
0006 F6 E3          MUL  BL      ;multiplicar
0008 8B D0          MOV  DX,AX   ;posicionar el producto
```

En la multiplicación con signo, el producto está en forma binaria real, si es positivo y, en forma de complemento a dos, si es negativo. Son las mismas formas empleadas para almacenar todos los números con signo positivo o negativo utilizados por el microprocesador. Si con el programa del ejemplo 4-13 se multiplican dos números con signo, sólo se cambia la instrucción MUL por IMUL.

TABLA 4–8 Instrucciones para la multiplicación de 8 bits

Instrucción	*Comentario*
MUL CL	Se multiplica AL por CL; esta multiplicación sin signo deja el producto en AX
IMUL DH	Se multiplica AL por DH; esta multiplicación con signo deja el producto en AX
IMUL BYTE PTR [BX]	Se multiplica AL por el contenido byte en la localidad de memoria del segmento de datos direccionada por BX; esta multiplicación sin signo deja el producto en AX
MUL TEMP	Se multiplica AL por el contenido de la localidad TEMP de la memoria en el segmento de datos; si TEMP se define como un número de 8 bits, el producto sin signo se encuentra en AX

TABLA 4–9 Instrucciones para la multiplicación de 16 bits

Instrucción	*Comentario*
MUL CX	Se multiplica AX por CX; el producto sin signo se encuentra en DX—AX
IMUL DI	Se multiplica AX por DI; el producto con signo se encuentra en DX—AX
MUL WORD PTR [SI]	Se multiplica AX por el contenido palabra de la localidad de memoria en el segmento de datos direccionada por SI; el producto sin signo se encuentra en DX—AX

Multiplicación de 16 bits. La multiplicación de palabras es muy semejante a la multiplicación de bytes. La diferencia es que el multiplicando está en AX en vez de AL y el producto aparece en DX–AX en lugar de AX. El registro DX siempre contiene los 16 bits más significativos del producto y AX los 16 bits menos significativos. Igual que con la multiplicación de 8 bits, el programador es quien elige el multiplicador. En la tabla 4-9 se presentan algunas instrucciones diferentes para la multiplicación de 16 bits.

Multiplicación inmediata de 16 bits. En los microprocesadores 8086/8088 no se podía efectuar multiplicación inmediata, pero en los 80286, 80386 y 80486 sí se puede con el empleo de una versión especial de la instrucción para multiplicar. La multiplicación inmediata debe tener signo y el formato de la instrucción es diferente porque contiene tres operandos. El primer operando es el registro destino de 16 bits; el segundo es un registro o localidad de la memoria que contiene el multiplicando de 16 bits; el tercero son datos inmediatos de 8 o de 16 bits utilizados como multiplicador.

La instrucción IMUL CX,DX,12H, multiplica 12H por DX y deja un producto de 16 bits con signo en CX. Si los datos inmediatos son de 8 bits, se extienden por signo a un número de 16 bits antes de que ocurra la multiplicación. Otro ejemplo es IMUL BX,NUMERO,1000H, que multiplica NUMERO por 1000H y deja el producto en BX. El destino y el multiplicando deben ser números de 16 bits. Aunque ésta es una multiplicación inmediata, las restricciones impuestas limitan su utilidad, en especial el hecho de que es una multiplicación con signo y que el producto tiene 16 bits de ancho.

Multiplicación de 32 bits. En los microprocesadores 80386 y 80486, se permite la multiplicación de 32 bits porque contienen registros de 32 bits. Igual que para la multiplicación de 8 y de 16 bits,

TABLA 4–10 Instrucciones para multiplicación de 32 bits

Instrucción	*Comentario*
MUL ECX	Se multiplica EAX por ECX; el producto sin signo se encuentra en EDX—EAX
IMUL EDI	Se multiplica EAX por EDI; el producto con signo se encuentra en EDX—EAX
MUL DWORD PTR[ECX]	Se multiplica EAX por el contenido doble palabra en la localidad de memoria en el segmento de datos direccionada por ECX; el producto sin signo se encuentra en EDX—EAX

se pueden multiplicar números con signo o sin signo, con el empleo de las instrucciones IMUL y MUL. En la multiplicación de 32 bits, el contenido de EAX se multiplica por el operando especificado con la instrucción. El producto, de 64 bits de ancho, se encuentra en EDX y en EAX, donde EAX contiene los 32 bits menos significativos del producto. En la tabla 4-10 aparecen algunas de las instrucciones de multiplicación de 32 bits que son parte del conjunto de instrucciones para los 80386 y 80486.

División

La división, igual que la multiplicación, se efectúa con números de 8 o de 16 bits en los 8086-80486 y también números de 32 bits en los 80386 y 80486. Estos números son enteros con signo (IDIV) o sin signo (DIV). El dividendo siempre es uno de doble ancho que se divide entre el operando. Esto significa que en una división de 8 bits se divide un número de 16 bits entre uno de 8 bits; en una división de 16 bits se divide un número de 32 bits entre uno de 16; en la división de 32 bits se divide un número de 64 bits entre uno de 32 bits. En los microprocesadores 8086-80486 no hay disponibles instrucciones para división inmediata.

Ninguno de los bits de bandera tiene un cambio predecible en una división. La división puede dar por resultado dos tipos diferentes de errores. Uno de ellos es el intento de dividir entre cero y, el otro, es un sobreflujo por división. El sobreflujo por división ocurre cuando se divide un número pequeño entre un número grande. Por ejemplo, supóngase que AX = 3000 y que se lo divide entre 2. Debido a que el cociente de una división de 8 bits aparece en AL, el resultado de 1500 ocasiona un sobreflujo de división porque el 1500 no cabe en AL. En ambos casos, el microprocesador genera una interrupción si ocurre un error en la división. La interrupción por error en la división y todas las demás instrucciones del microprocesador se explican en el capítulo 5.

División de 8 bits. Una división de 8 bits utiliza el registro AX para almacenar el dividendo, que se divide entre el contenido de cualquier registro o localidad de memoria de 8 bits. El cociente se transfiere a AL después de la división y AH contiene un residuo de número entero. Para una división con signo, el cociente es positivo o negativo, pero el residuo es siempre un entero positivo. Por ejemplo, si AX = 0010H (+16) y BL = FDH (-3) y se ejecuta la instrucción IDIV BL, entonces AX = 01FBH. Esto representa un cociente de -5 (AL) y un residuo de 1 (AH). En la tabla 4-11 se presentan algunas de las instrucciones para la división de 8 bits.

Para la división de 8 bits, los números, por lo general, tienen 8 bits de ancho. Esto significa que uno de ellos, el dividendo, se debe convertir a un número de 16 bits de ancho en AX, lo cual

TABLA 4–11 Instrucciones para división de 8 bits

Instrucción	*Comentario*
DIV CL	Se divide AX entre CL; el cociente sin signo se encuentra en AL y el residuo en AH
IDIV BL	Se divide AX entre BL; el cociente con signo está en AL y el residuo en AH
DIV BYTE PTR [BP]	Se divide AX entre el contenido de bytes de la localidad en la memoria en el segmento de pila direccionada por BP; el cociente sin signo se encuentra en AL y el residuo en AH

se logra en una forma distinta para los números con signo y sin signo. Para los números sin signo, los 8 bits más significativos deben ser cero (extensión por cero). La instrucción MOVZX descrita en el capítulo 3 se puede emplear para extender ceros un número en los microprocesadores 80386 y 80486. Para el número con signo, los 8 bits menos significativos se extienden por signo dentro de los 8 bits más significativos. En el microprocesador hay una instrucción especial que extiende por signo a AL dentro de AH o convierte un número de 8 bits con signo en AL en un número de 16 bits con signo en AX. La instrucción CBW *(convertir byte a palabra)* efectúa esta conversión. En los microprocesadores 80386 y 80486 una instrucción MOVSX (véase capítulo 4) puede extender o ampliar por signo un número.

EJEMPLO 4–14

```
0000 A0 0000 R          MOV  AL,NUMB        ;obtener NUM
0003 B4 00              MOV  AH,0           ;extender con ceros
0005 F6 36 0002 R       DIV  NUMB1          ;dividir entre NUM1
0009 A2 0003 R          MOV  ANSQ,AL        ;salvar el cociente
000C 88 26 0004 R       MOV  ANSR,AH        ;salvar el residuo
```

En el ejemplo 4-14 se ilustra un programa corto para dividir el contenido de bytes sin signo de la localidad NUM en la memoria, entre el contenido sin signo de la localidad NUM1 de la memoria. En este caso, se almacena el cociente en la localidad RESPC y el residuo en la localidad RESPR. Se debe tener en cuenta la forma en que el contenido de la localidad NUM se recupera de la memoria y luego, se extiende con ceros para formar un número de 16 bits sin signo para el dividendo.

EJEMPLO 4–15

```
0000 A0 0000 R          MOV   AL,NUM        ;obtener NUM
0003 98                 CBW                 ;extender por signo
0004 F6 3E 0002 R       IDIV  NUM1          ;dividir entre NUM1
0008 A2 0003 R          MOV   RESPC AL      ;salvar el cociente
000B 88 26 0004 R       MOV   RESPR AH      ;salvar el residuo
```

En el ejemplo 4-15 se presenta el mismo programa básico, excepto que los números tienen signo. Esto significa que, en lugar de extender con ceros a AL en AH, se extiende con signo con el empleo de la instrucción CBW.

División de 16 bits. La división de 16 bits es semejante a la división de 8 bits, excepto que en lugar de dividir entre AX un dividendo de 32 bits se divide entre DX-AX. El cociente aparece en AX y el residuo en DX después de una división de 16 bits. En la tabla 4-12 se presentan algunas de las instrucciones para la división de 16 bits.

Igual que para la división de 8 bits, hay que convertir los números a la forma adecuada para el dividendo. Si se empieza con un número de 16 bits sin signo en AX, entonces DX se debe «borrar» a 0. En los 80386 y 80486, el número se extiende con ceros con el empleo de la instrucción MOVZX. Si AX es un número de 16 bits con signo, la instrucción CWD *(convertir palabra a doble palabra)*, lo extiende por signo a formar un número de 32 bits con signo. Si se tiene un microprocesador 80386 u 80486, también se puede emplear la instrucción MOVSX para extender un número por signo.

En el ejemplo 4-16 se ilustra la división de dos números de 16 bits con signo. En este caso, se divide un −100 en AX entre un +9 en CX. La instrucción CWD convierte el −100 en AX a un

TABLA 4–12 Instrucciones para división de 16 bits

Instrucción	Comentario
DIV CX	Se divide DX—AX entre CX; el cociente sin signo está en AX y el residuo en DX
IDIV SI	Se divide DX—AX entre SI; el cociente con signo está en AX y el residuo en DX
DIV NUMB	Se divide DX—AX entre el contenido de palabras en la localidad NUM de la memoria en el segmento de datos; el cociente sin signo está en AX y el residuo está en DX

−100 en AX-DX antes de la división; después, el resultado aparece en DX-AX como cociente de −11 en AX y un residuo de 1 en DX.

EJEMPLO 4–16

```
0000 B8 FF9C          MOV   AX,-100        ;cargar-100
0003 B9 0009          MOV   CX,9           ;cargar +9
0006 99               CWD                  ;extender por signo
0007 F7 F9            IDIV  CX
```

División de 32 bits. Los microprocesadores 80386 y 80486 efectúan divisiones de 32 bits en números con signo o sin signo. El contenido de 64 bits de EDX—EAX se divide entre el operando especificado por la instrucción y queda un cociente de 32 bits en EAX y un residuo de 32 bits en EDX. Aparte del tamaño de los registros, esta instrucción funciona igual que las divisiones de 8 y de 16 bits. En la tabla 4-13 se presentan algunos ejemplos de instrucciones para división de 32 bits. La instrucción convertir doble palabra a cuádruple palabra (CDQ) se utiliza antes de una división con signo para convertir el contenido de 32 bits de EAX a un número de 64 bits con signo en EDX—EAX.

El residuo. ¿Qué se hace con el residuo después de una división? Hay cuántas opciones posibles. Se podría utilizar el residuo para redondear el resultado o se podría eliminar el residuo para

TABLA 4–13 Instrucciones para la división de 32 bits

Instrucción	Comentario
DIV ECX	Se divide EDX—EAX entre ECX; el cociente sin signo está en EAX y el residuo, en EDX
DIV DATO2	Se divide EDX—EAX entre el contenido doble palabra en la localidad DATO2 de la memoria en el segmento de datos; el cociente sin signo está en EAX y el residuo está en EDX
IDIV DWORD PTR [EDI]	Se divide EDX—EAX entre el contenido doble palabra de la localidad en la memoria del segmento de datos direccionada por EDI; el cociente con signo está en EAX y el residuo está en EDX

truncar el resultado. Si la división es de números sin signo, el redondeo requiere comparar el residuo con la mitad del divisor para decidir si se redondea o no el cociente. También se podría convertir el residuo a uno fraccional.

EJEMPLO 4–17

```
0000 F6 F3              DIV  BL         ;dividir
0002 02 E4              ADD  AH,AH      ;duplicar el residuo
0004 3A E3              CMP  AH,BL      ;probar si hay redondeo
0006 72 02              JB   NEXT
0008 FE C0              INC  AL         ;redondear
000A          NEXT:
```

En el ejemplo 4-17 se muestra una secuencia de instrucciones para dividir AX entre BL y redondear el resultado. Este programa duplica el residuo antes de compararlo con DL para decidir si se redondea o no el cociente. En este caso, una instrucción INC redondea el contenido de AL después de la comparación.

Supóngase que se necesita un residuo en fracciones en lugar de un residuo de enteros. Para obtener el residuo en fracciones se salva el cociente. Después se «borra» a cero el registro AL. Ahora, el número restante en AX se divide entre el operando original para generar un residuo en fracciones.

EJEMPLO 4–18

```
0000 B8 000D            MOV  AX,13      ;cargar 13
0003 B3 02              MOV  BL,2       ;cargar 2
0005 F6 F3              DIV  BL         ;13/2
0007 A2 0003 R          MOV  RESPC AL   ;salvar el cociente
000A B0 00              MOV  AL,0       ;borrar AL
000C F6 F3              DIV  BL         ;generar el residuo
000E A2 0004 R          MOV  RESPR AL   ;salvar el residuo
```

En el ejemplo 4-18 se muestra cómo se divide un 13 entre un 2. El cociente de 8 bits se salva en la localidad RESPC de la memoria y se «borra» AL. A continuación, se divide otra vez el contenido de AX entre 2 para generar un residuo fraccional. Después de la división, el registro AH es igual a 80H, o sea un 100000002. Si el punto binario (base) se coloca antes del bit de AL que está más hacia la izquierda, se tendrá un residuo fraccional de 0.10000000_2 o un 0.5 decimal. En este ejemplo, el residuo se salva en la localidad RESPR de la memoria.

4–3 ARITMETICA PARA BCD Y ASCII

Los microprocesadores 8086-80486 permiten el manejo aritmético de datos decimales codificados en binario (BCD) o el Código «estándar» norteamericano para intercambio de información (ASCII por sus siglas en inglés). Esto se logra con una instrucción que ajusta los números para la aritmética BCD y ASCII.

Las operaciones en BCD ocurren en los sistemas utilizados, por ejemplo, en registradoras en establecimientos de venta y en otros sistemas que rara vez requieren aritmética. Las operaciones

con ASCII se efectúan en datos de ese código utilizado por algunos programas. En la actualidad, rara vez se utiliza aritmética BCD o ASCII.

Aritmética para BCD

Hay dos técnicas aritméticas que funcionan con datos de BCD: suma y resta. El conjunto de instrucciones da dos instrucciones que corrigen el resultado de una suma y de una resta de BCD. La instrucción DAA *(ajuste decimal después de la suma)* va después de la suma de BCD y DAS *(ajuste decimal después de la resta)* después de la resta. Ambas instrucciones corrigen el resultado de la suma o la resta, por lo cual es un número BCD.

Para los datos de BCD los números siempre aparecen en la forma de BCD empacado y se almacenan como dos dígitos BCD por byte. Las instrucciones para ajustar sólo funcionan con el registro AL, después de la suma y resta en BCD.

Instrucción DAA. La instrucción DAA va después de la instrucción ADD o ADC para ajustar el resultado de manera que sea BCD. Supóngase que DX y BX contiene, cada uno, números de 4 dígitos empacados de BCD. En el ejemplo 4-9 se presenta un programa corto que suma los números BCD en DX y BX y almacena el resultado en CX.

EJEMPLO 4–19

```
0000 BA 1234        MOV  DX,1234H       ;cargar 1,234
0003 BB 3099        MOV  BX,3099H       ;cargar 3,099
0006 8A C3          MOV  AL,BL          ;sumar BL con CL
0008 02 C2          ADD  AL,DL
000A 27             DAA                 ;ajustar
000B 8A C8          MOV  CL,AL          ;respuesta a CL
000D 8A C7          MOV  AL,BH          ;sumar BH con DH y CF
000F 12 C6          ADC  AL,DH
0011 27             DAA                 ;ajustar
0012 8A E8          MOV  CH,AL          ;respuesta a CH
```

Debido a que la instrucción DAA sólo funciona con el registro AL, esta suma debe ocurrir a razón de 8 bits cada vez. Después de sumar los registros BL y DL, se ajusta al resultado con una instrucción DAA antes de almacenarlo en CL. Luego, se suman los registros BH y DH con acarreo y se vuelve a ajustar el resultado con DAA antes de almacenarlo en CH. En este ejemplo, se suma un 1,234 a un 3,099 para generar una suma de 4,333 que se transfiere a CX después de la suma. Se debe tener en cuenta que 1234 BCD es lo mismo que 1234H.

EJEMPLO 4–20

```
0000 BA 1234        MOV  DX,1234H       ;cargar 1,234
0003 BB 3099        MOV  BX,3099H       ;cargar 3,099
0006 8A C3          MOV  AL,BL          ;restar DL de BL
0008 2A C2          SUB  AL,DL
000A 2F             DAS                 ;ajustar
000B 8A C8          MOV  CL,AL          ;respuesta a CL
000D 8A C7          MOV  AL,BH          ;restar DH y CF, de BH
000F 1A C6          SBB  AL,DH
0011 2F             DAS                 ;ajustar
0012 8A E8          MOV  CH,AL          ;respuesta a CH
```

Instrucción DAS. La instrucción DAS funciona como la instrucción DAA excepto que va después de una resta, en vez de una suma. El ejemplo 4-20 es básicamente el mismo que el ejemplo 4-19, excepto que resta DX y BX en vez de sumarlos. La diferencia principal en estos programas es que las instrucciones DAA cambian a DAS y las instrucciones ADD y ADC cambian a SUB y SBB.

Aritmética para ASCII

Las instrucciones aritméticas para ASCII funcionan con números codificados en ASCII. El valor de estos números es entre 30H y 39H para los números 0 a 9. En las operaciones aritméticas con ASCII se utilizan cuatro instrucciones: AAA (*ajuste ASCII después de la suma*), AAD (*ajuste ASCII antes de la división*), AAM (*ajuste ASCII después de la multiplicación*) y AAS (*ajuste ASCII después de la resta*). En estas instrucciones se utiliza el registro AX como fuente y como destino.

Instrucción AAA. La suma de dos números de un dígito codificados en ASCII no dará por resultado ningún dato útil. Por ejemplo, si se suman un 31H y un 39H, el resultado es un 6AH. Esta suma en ASCII (1 + 9) produciría un resultado ASCII de dos dígitos, equivalente a un 10 decimal, el cual es un 31H y un 30H en código ASCII. Si se ejecuta la instrucción AAA después de esta suma, el registro AX contendrá un 0100H que, aunque no es un código ASCII, se puede convertir si se le suma 3030H, que genera 3130H. La instrucción AAA borra AH si el resultado es menor de 10 y agrega un 1 a AH si el resultado es mayor de 10.

EJEMPLO 4–21

```
0000 B8 0031          MOV  AX,31H        ;cargar ASCII 1
0003 04 39            ADD  AL,39H        ;sumar ASCII 9
0005 37               AAA                ;ajustar
0006 05 3030          ADD  AX,3030H      ;responder a ASCII
```

En el ejemplo 4-21 se muestra cómo funciona esta suma ASCII en el microprocesador. Se debe tener en cuenta que se emplea la instrucción MOV AX,31H para «borrar» AH antes de la suma. El operando de 0031H coloca un 00H en AH y un 31H en AL.

Instrucción AAD. La instrucción AAD, al contrario de todas las otras instrucciones para ajuste, aparece antes de la división. La instrucción AAD requiere que el registro AX contenga un número BCD no empacado, de 2 dígitos (no ASCII) antes de que se ejecute. Después de ajustar el registro AX con AAD se divide entre un número BCD no empacado para generar un resultado de un solo dígito en AL y cualquier residuo quedará en AH.

EJEMPLO 4–22

```
0000 B3 09            MOV  BL,9          ;cargar divisor
0002 B8 0702          MOV  AX,0702H      ;cargar dividendo
0005 D5 0A            AAD                ;ajustar
0007 F6 F3            DIV  BL
```

En el ejemplo 4-22 se ilustra la forma en que un 72 en BCD sin empacar se divide entre 9 para producir un cociente de 8. La instrucción AAD ajusta el 0702H cargado en el registro AX a 0048H. Se debe tener en cuenta que esto convierte a un número BCD sin empacar, de 2 dígitos, en un número binario, a fin de poder dividirlo con la instrucción (DIV) para división binaria. La instrucción AAD convierte los números BCD no empacados, entre 00 y 99, a binario.

Instrucción AAM. La instrucción AAM sigue a la instrucción para multiplicación, después de multiplicar dos números de un dígito en BCD no empacados. En el ejemplo 4-23 se presenta un programa corto para multiplicar un 5 por un 5; el resultado después de la multiplicación es 0018H en el registro AX. Después de ajustar el resultado con la instrucción AAM, AX contiene un 0205H, que es un resultado de 25 en BCD no empacado. Si se suma 3030H a 0205H, se convierte en un resultado en ASCII 3235H.

EJEMPLO 4–23

```
0000 B0 05          MOV  AL,5          ;cargar multiplicando
0002 B1 05          MOV  CL,5          ;cargar multiplicador
0004 F6 E1          MUL  CL
0006 D4 0A          AAM                ;ajustar
```

Un beneficio adicional de la instrucción AAM es que convertirá de binario a BCD no empacado. Si aparece un número binario entre 0000H y 0063H en el registro AX, la instrucción AAM lo convierte a BCD. Por ejemplo, si AX contiene un 0060H antes de ejecutar AAM, contendrá un 0906H después de que se ejecute, que es el equivalente de 96 en BCD no empacado. Si se suma 3030H a 0906H, se acabará de cambiar el resultado a ASCII.

EJEMPLO 4–24

```
0000 33 D2          XOR   DX,DX        ;borrar registro DX
0002 B9 0064        MOV   CX,100       ;dividir DX-AX entre 100
0005 F7 F1          DIV   CX           ;AX=cociente, DX=residuo
0007 D4 0A          AAM                ;convertir cociente a BCD
0009 05 3030        ADD   AX,3030H     ;convertir a ASCII
000C 92             XCHG  AX,DX        ;repetir para el residuo
000D D4 0A          AAM
000F 05 3030        ADD   AX,3030H
```

En el ejemplo 4-24 se muestra la forma en que el contenido binario de 16 bits de AX se convierte a una cadena de caracteres en ASCII de 4 dígitos con el empleo de la división y de la instrucción AAM. Se debe tener en cuenta que esto funciona para números entre 0 y 9999. Primero, se «borra» DX y luego se divide DX-AX entre 100. Por ejemplo, si AX =245 después de la división, AX = 2 y DX = 45. Estas mitades separadas se convierten a BCD con el empleo de AAM, y luego se suma un 3030H para convertir al código ASCII.

Instrucción AAS. Esta instrucción, igual que otras instrucciones para ajuste ASCII, ajusta el registro AX después de una resta. Por ejemplo, supóngase que se resta un 35H de un 39H. El resultado será un 04H, que no requiere corrección; en este caso AAS no modificará ni a AH ni a AL. Por otra parte, si se resta 38H de 37H, entonces AL será igual a 09H y el número en AH se decrementará en 1. Este decremento permite restar entre los números ASCII de múltiples dígitos.

4–4 INSTRUCCIONES LOGICAS BASICAS

Las instrucciones lógicas básicas incluyen AND, OR, OR exclusivo y NOT. Otra instrucción lógica es TEST que se explica en esta sección, porque es una forma especial de la instrucción AND. Además, se explica la instrucción NEG, que es similar a la instrucción NOT.

Las operaciones lógicas proporcionan el control de bits binarios en programación de «*bajo nivel*». Las instrucciones lógicas permiten hacer uno, cero o complementar bits. La programación de bajo nivel aparece en forma de lenguaje de máquina o lenguaje de ensamblador y a menudo controla los dispositivos de E/S en un sistema. Todas las instrucciones lógicas afectan los bits de bandera. Las operaciones lógicas siempre hacen cero las banderas de acarreo y sobreflujo, mientras que las otras banderas cambian para reflejar la condición del resultado.

Cuando se «manipulan» los datos binarios de un registro o de una localidad de memoria, la posición que está más a la derecha siempre se numera bit 0. Los números de posición de los bits aumentan desde 0 hacia la izquierda hasta el 7 para un byte y hasta 15 para una palabra. En una doble palabra (32 bits) se utiliza la posición 31 como el bit que está más a la izquierda.

AND

La operación AND efectúa una multiplicación lógica como se ilustra en la tabla de la verdad en la figura 4-3. En este caso, se ejecuta la operación AND con dos bits, A y B para producir el resultado X. Como se indica en la tabla de la verdad, X es un 1 lógico sólo cuando A y B son también 1 lógico. Para todas las demás combinaciones de entrada, de A y B, X es un cero lógico. Es importante recordar que 0 AND cualquier cosa es siempre 0 y que 1 AND 1 es siempre un 1.

La instrucción AND A menudo sustituye a las compuertas AND discretas si la velocidad requerida no es muy grande. En los microprocesadores 8086-80486, la instrucción AND, a menudo se ejecuta en un microsegundo. Por ello, si el circuito al cual sustituye la instrucción AND funciona con un retardo mayor de un microsegundo, la instrucción AND es el sustituto más razonable. Esta sustitución ahorra mucho dinero. Un solo circuito integrado (7408) para una compuerta AND, cuesta alrededor de 40 centavos de dólar en Estados Unidos, mientras que cuesta menos de una centésima de centavo (0.01) almacenar la instrucción AND en una memoria de sólo lectura (ROM).

La operación AND también «borra» los bits de un número binario. La tarea de borrar o hacer cero un bit en un número binario, se llama *enmascaramiento*. En la figura 4-4 se ilustra el proceso. Se verá que los cuatro bits que están más a la izquierda se hacen a cero, ya que 0 AND cualquier cosa, es 0. Las posiciones de los bits en los que se ejecuta el AND y 1, no cambian. Esto ocurre porque si se ejecuta la instrucción AND con un 1, el resultado es 1 y si se ejecuta con un 0, el resultado es cero.

FIGURA 4–3 Tabla de la verdad para demostrar la operación lógica AND.

A ·	B =	X
0	0	0
0	1	0
1	0	0
1	1	1

XXXX	XXXX	(Patrón incógnito)
· 0000	1111	(Patrón de mascarilla)
0000	XXXX	(Resultado)

FIGURA 4–4 Una máscara (0000 1111) se ejecuta en un AND con un número incógnito. Se verá que los 4 bits más a la izquierda se enmascaran (borran) a 0.

La instrucción AND utiliza cualquier modo de direccionamiento, excepto memoria a memoria y registro de segmentos. En la tabla 4-14 aparece una lista de algunas de las instrucciones AND y un comentario de su operación funcionamiento.

EJEMPLO 4–25

```
0000 BB 3135           MOV  BX,3135H        ;cargar ASCII
0003 81 E3 0F0F        AND  BX,0F0FH        ;enmascarar BX
```

Un número codificado en ASCII se puede convertir a BCD con el empleo de la instrucción AND para enmascarar las posiciones más a la izquierda de 4 bits binarios. Con ello, se convierten los números ASCII 30H a 39H, a 0-9. En el ejemplo 4-25 se presenta un programa corto para convertir el contenido ASCII de BX a BCD. La instrucción AND en este ejemplo es para convertir en forma simultánea 2 dígitos, de ASCII a BCD.

OR

La operación OR efectúa suma lógica y a menudo se denomina *Función OR inclusiva*. La función OR genera un 1 de salida si cualquiera de las entradas es 1. En la salida aparece un 0, sólo cuando

TABLA 4–14 Instrucción AND

Instrucción	*Comentario*
ADN AL,BL	AL = AL AND BL
ADN CX,DX	CX = CX AND DX
AND ECX,EDI	ECX = ECX AND EDI
AND CL,33H	CL = CL AND 33H
AND DI,4FFFH	DI = DI AND 4FFFH
AND ESI,34H	ESI = ESI AND 00000034H
AND AX,[DI]	Se ejecuta la función AND entre AX y el contenido palabra de la localidad de la memoria del segmento de datos direccionado por DI; el resultado se transfiere a AX
AND ARREGLO [SI],AL	Se ejecuta el AND lógico entre el contenido byte de la localidad de la memoria en el segmento de datos, direccionado por ARREGLO + SI y AL; el resultado se transfiere a la memoria
AND [EAX],CL	Se ejecuta el AND lógico entre el contenido byte de CL y el contenido byte de la localidad de la memoria en el segmento de datos direccionada por el registro EAX; el resultado se transfiere a la memoria

A	+ B	= X
0	0	0
0	1	1
1	0	1
1	1	1

FIGURA 4-5 Tabla de verdad para ilustrar la operación OR (suma lógica). Se verá que se emplea signo positivo (+) para indicar una operación con OR.

	XXXX	XXXX	(Patrón incógnito)
+	0000	1111	(Patrón de enmascaramiento)
	XXXX	1111	(Resultado)

FIGURA 4-6 Un patrón para prueba (0000 1111) se somete a suma lógica con un número incógnito, para indicar que la operación OR se emplea para el establecimiento selectivo de bits.

todas las entradas son 0. La tabla de verdad para la función OR aparece en la figura 4-5. En este caso, se ejecuta la instrucción OR en las entradas A y B para producir la salida X. Es importante recordar que 1 OR cualquier cosa, produce un 1.

La instrucción OR, a menudo sustituye a las compuertas discretas OR, lo cual produce un considerable ahorro, porque una compuerta OR cuádruple, de 2 entradas (7432) cuesta alrededor de 40 centavos de dólar (en Estados Unidos) mientras que almacenar en una memoria sólo de lectura una instrucción OR cuesta menos de una centésima de centavo (0.01).

En la figura 4-6 se ilustra la forma en que la compuerta OR hace 1 cualquier bit de un número binario. En este caso, en un número desconocido (XXXX XXXX) se ejecuta una instrucción OR con un 0000 1111 para dar un resultado de XXXX 1111. Se hace el grupo de 4 bits que están más a la derecha, mientras que los cuatro bits que están más a la izquierda siguen sin cambio. La operación OR hace uno cualquier bit y la operación AND lo hace cero.

Con la instrucción OR se utiliza cualquiera de los modos de direccionamiento permitidos para cualquier otra instrucción, excepto el direccionamiento de registro de segmento. En la tabla 4-15 aparecen algunos ejemplos de la instrucción y comentarios de su operación.

Supóngase que se multiplican dos números BCD y se ajustan con la instrucción AMM. El resultado aparece en AX como número de 2 dígitos de BCD sin empacar. En el ejemplo 4-26 se ilustra la multiplicación y la forma de cambiar el resultado a un número de 2 dígitos codificados en ASCII con la instrucción OR. En este caso, OR AX, 3030H convierte en 3335H al 0305H que hay en AX. La instrucción OR se puede sustituir con una ADD AX, 3030H para obtener los mismos resultados.

TABLA 4–15 Instrucción OR

Instrucción	*Comentario*
OR AH,BL	AH = AH OR BL
OR SI,DX	SI = SI OR DX
OR EAX,EBX	EAX = EAX OR EBX
OR DH,0A3H	DH = DH OR 0A3H
OR SP,990DH	SP = SP OR 990DH
OR EBP,10	EBP = EBP OR 0000000AH
OR DX,[BX]	Se ejecuta el OR lógico entre DX y el contenido palabra de la localidad de la memoria en el segmento de datos direccionado por BX
OR DATOS [DI + 2],AL	Se ejecuta el OR lógico entre la dirección DATOS y DI +2, de la memoria en el segmento de datos y AL

EJEMPLO 4–26

```
0000 B0 05        MOV  AL,5        ;cargar datos
0002 B3 07        MOV  BL,7
0004 F6 E3        MUL  BL
0006 D4 0A        AAM              ;ajustar
0008 0D 3030      OR   AX,3030H    ;a ASCII
```

OR Exclusivo

La instrucción Or Exclusivo (XOR) es diferente que la instrucción OR Inclusivo (OR). La diferencia es que la condición de 1,1 en la función OR produce un 1, mientras que la condición de 1,1 de la función OR exclusivo, produce un 0. La instrucción OR Exclusivo excluye esa condición, pero la OR Inclusivo sí la incluye.

En la figura 4-7 se muestra la tabla de la verdad de la función Or Exclusivo. (Compárela con la figura 4-5 para apreciar la diferencia entre estas dos funciones OR.) Si ambas entradas de la

FIGURA 4–7 La tabla de verdad para la función OR exclusivo (XOR). Se verá que se emplea el signo % para indicar una operación OR Exclusivo.

A	⊕ B	= X
0	0	0
0	1	1
1	0	1
1	1	0

TABLA 4-16 Instrucciones para OR Exclusivo (XOR)

Instrucción	*Comentario*
XOR CH,DL	CH = CH XOR DL
XOR SI,BX	SI =SI XOR BX
XOR EBX,EDI	EBX = EBX XOR EDI
XOR CH,0EEH	CH = CH XOR 0EEH
XOR DI,00DDH	DI = DI XOR 00DDH
XOR ESI,100	ESI = ESI XOR 00000064H
XOR DX,[SI]	Se ejecuta el XOR DX y el contenido palabra de la localidad de la memoria en el segmento de datos direccionado por SI: el resultado se deja en DX
XOR DATES[DI + 2],AL	Se ejecuta el XOR entre el contenido byte de la localidad de la memoria del segmento de datos direccionado por DATOS más DI + 2 y AL; el resultado se deja en la memoria

función OR exclusivo son 0 o 1, la salida es 0. Si las entradas son diferentes, la salida es un 1. Debido a ello, a la instrucción OR exclusivo se le llama, a veces, *comparador*.

En la instrucción XOR se utiliza cualquier modo de direccionamiento, excepto el direccionamiento de registro de segmento. En la tabla 4-16 se presentan diversas formas de la instrucción OR Exclusivo y comentarios de su funcionamiento.

Igual que las funciones AND y OR, la OR Exclusivo también sustituye a circuitos lógicos discretos. La puerta OR Exclusivo, cuádruple, de 2 entradas (7486) se sustituye con una instrucción XOR. La 7486 cuesta alrededor de 40 centavos de dólar mientras que almacenar la instrucción en la memoria cuesta menos de una centésima de centavo. Al sustituir, aunque sea una sola 7486, se ahorra mucho dinero, en especial si se construyen muchos sistemas.

La instrucción OR Exclusivo es útil si hay que invertir algunos bits de un registro o localidad de la memoria. Esta instrucción permite invertir o complementar una parte de un número. En la figura 4-8 se ilustra la forma en que una parte de una cantidad incógnita se puede invertir con XOR. Se verá que cuando se ejecuta un OR Exclusivo de 1, con X, el resultado es $\overline{X}$. Si se ejecuta un OR Exclusivo de 0 actúa con X, el resultado es X.

Supóngase que se deben invertir los 10 bits que están más a la izquierda en el registro BX sin cambiar los 6 bits que están más a la derecha. Esta tarea se logra con la instrucción XOR BX, OFFC0H. La instrucción AND «borra» (hace 0) los bits, la instrucción OR «activa» (hace 1) los bits y la instrucción OR Exclusivo invierte los bits. Estas tres instrucciones permiten que un

	XXXX	XXXX	(Patrón incógnito)
⊕	0000	1111	(Patrón de prueba)
	XXXX	$\overline{XXXX}$	(Resultado)

FIGURA 4-8 Un patrón de prueba (0000 1111) operado con OR Exclusivo con una cantidad incógnita produjo inversión en las posiciones de bit en donde el patrón de prueba contiene 1 lógico.

TABLA 4–17 Instrucciones TEST

Instrucción	*Comentario*
TEST DL,DH	Se ejecuta el AND lógico entre DL y DH; no cambian DL ni DH; sólo cambian las banderas
TEST CX,BX	Se ejecuta el AND lógico entre CX y BX; no cambian CX ni BX; sólo cambian las banderas
TEST EDX.ECX	Se ejecuta el AND lógico entre EDX y ECX; no cambian EDX ni ECX; sólo cambian las banderas
TEST AH,4	Se ejecuta el AND lógico entre AH y 4; AH no cambia; sólo cambian las banderas
TEST EAX,256	Se ejecuta el AND lógico EAX con AND con 256; EAX no cambia; sólo cambian las banderas

programa tenga el control completo de cualquier bit, almacenado en cualquier registro o localidad de la memoria. Estas operaciones lógicas son ideales para aplicaciones en sistemas de control en donde hay que conectar o poner a funcionar (1) el equipo o apagarlo (0) y cambiarlo de encendido a apagado o viceversa.

Instrucciones TEST y Bit Test

La instrucción TEST efectúa la operación AND. La diferencia es que la instrucción AND cambia el operando destino, pero la instrucción TEST no lo hace. TEST sólo afecta la condición del registro de banderas, que indica el resultado de la prueba. En la instrucción TEST se emplean los mismos modos de direccionamiento que en la instrucción AND. En la tabla 4-17 se ilustran algunas formas de la instrucción TEST con comentarios de funcionamiento.

La instrucción TEST funciona igual que la instrucción CMP. La diferencia es que la TEST, por lo general, sólo prueba un bit, mientras que la instrucción CMP prueba todo el byte o la palabra. La bandera de cero (Z) es un 1 lógico (que indica un resultado de cero) si el bit que se prueba es un cero y Z = 0 (que indica un resultado que no es cero) si el bit que se prueba no es cero.

Por lo general, la instrucción TEST va seguida por una instrucción JZ (brincar si es cero) y JNZ (brincar si no es cero). El operando destino se suele probar contra los datos inmediatos. El valor de los datos inmediatos es 1, para probar la posición del bit más a la derecha o un 2 para probar el siguiente bit, 4 para el que sigue, etcétera.

EJEMPLO 4–27

```
000  A8 01        TEST  AL,1        ;probar bit derecho
0002 75 1C        JNZ   ;DER        ;si es uno
0004 A8 80        TEST  AL,128      ;probar bit izquierdo
0006 75 38        JNZ   ;IZQ        ;si es uno
```

En el ejemplo 4-27 se presenta un programa corto para probar las posiciones de los bits más a la derecha y más a la izquierda del registro AL. En este caso, un 1 selecciona el bit más a la derecha y 128 selecciona el que está más hacia la izquierda. La instrucción JNZ que sigue a cada

TABLA 4–18 Instrucciones para probar bits

Instrucción	*Comentario*
BT	Prueba un bit en el operando destino especificado por el operando fuente
BTC	Prueba y complementa un bit en el operando destino especificado por el operando fuente
BTR	Prueba y hace cero un bit en el operando destino especificado por el operando fuente
BTS	Prueba y hace uno un bit en el operando destino especificado por el operando fuente

TEST permite brincar a diferentes localidades la memoria, según sea el resultado de las pruebas. La instrucción JNZ salta a la dirección del operando (o sea Derecha o Izquierda en el ejemplo) si el bit que se prueba no es cero.

Los microprocesadores 80386 y 80486 incluyen instrucciones adicionales para la prueba de los bits. En la tabla 4-18 se presentan las cuatro instrucciones Bit Test (prueba de bits) disponibles para esos microprocesadores.

Las cuatro formas de la instrucción Bit Test prueban la posición de bits en el operando destino, seleccionado por el operando fuente. Por ejemplo, la instrucción BT AX,4 prueba la posición 4 de bit en AX. El resultado de la prueba se coloca en el bit de la bandera de acarreo; pero, si la posición 4 de bit es un 1, se activa (1) el acarreo; si la posición 4 es 0, se desactiva (0) el acarreo.

Las tres instrucciones restantes para prueba de bits también colocan al bit sometido a prueba en la bandera de acarreo y, después, cambian a ese bit. La instrucción BTC AX,4 complementa la posición 4 de bit después de probarla; la instrucción BTR AX,4 la desactiva (0) después de la prueba; la instrucción BTS AX,4 la activa (1) después de la prueba.

NOT y NEG

La inversión lógica o complemento a uno (NOT) o la inversión aritmética con signo o complemento a dos (NEG) son las dos últimas funciones lógicas que se presentan, excepto corrimientos y rotaciones en la siguiente sección. Estas instrucciones son dos de las pocas que contienen un solo operando. En la tabla 4-19 se presentan algunas variantes de las instrucciones NOT y NEG. Igual que la mayor parte de las otras instrucciones, NOT y NEG pueden utilizar cualquier modo de direccionamiento, excepto direccionamiento de registro de segmento.

La instrucción NOT invierte todos los bits de un byte o una palabra. La instrucción NEG ejecuta el complemento a dos de un número, lo cual significa que el signo del número cambia de positivo a negativo o viceversa. La función NOT se considera lógica y la función NEG se considera una operación aritmética.

4–5 CORRIMIENTOS Y ROTACIONES

Las instrucciones para corrimiento y rotación manejan los números binarios a «nivel» de bit binario como lo hacen las instrucciones AND, OR, OR Exclusivo y NOT. Los corrimientos y

TABLA 4-19 Instrucciones NOT y NEG

Instrucción	*Comentario*
NOT CH	CH se complementa a uno
NEG CH	CH se complementa a dos
NEG AX	AX se complementa a dos
NOT EBX	EBX se complementa a uno
NEG ECX	ECX se complementa a dos
NOT TEMP	El contenido de la localidad la memoria del segmento de datos direccionado por TEMP se complementa a uno. El tamaño de TEMP se determina por la forma en que se lo defina
NOT BYTE PTR [BX]	El contenido bytes de la localidad de la memoria en el segmento de datos direccionada por BX se complementa a uno

rotaciones tienen máxima aplicación en la programación de «bajo nivel» utilizada para controlar los dispositivos de E/S. Los microprocesadores 80386 y 80486 contienen un conjunto completo de instrucciones de corrimiento y rotación para actuar en cualesquiera datos, registros o memoria.

Corrimientos

Las instrucciones para corrimiento colocan o mueven números a izquierda o derecha dentro de un registro o localidad de memoria. También efectúan operaciones aritméticas sencillas como multiplicación por potencias de 2^{+n} (*corrimiento a la izquierda*) o división entre potencias de 2^{-n} (*corrimiento a la derecha*). El conjunto de instrucciones para el 80286 contiene cuatro diferentes instrucciones para corrimiento: dos son corrimientos lógicos y dos son corrimientos aritméticos. Las cuatro operaciones de corrimiento aparecen en la figura 4-9.

Se debe tener en cuenta que en la figura 4-9 hay dos corrimientos diferentes a la derecha y dos diferentes a la izquierda. Los corrimientos lógicos mueven un 0 a la posición de bits más a la derecha para un corrimiento lógico a la izquierda y un 0 a la posición de bit más a la izquierda para un corrimiento lógico a la derecha. Hay también dos corrimientos aritméticos, que son idénticos a los corrimientos lógicos. Los corrimientos aritméticos y lógicos a la derecha son diferentes porque el corrimiento aritmético a la derecha copia el bit de signo en el número, mientras que el corrimiento lógico a la derecha copia un 0 en el número.

Los operaciones de corrimiento lógico a la derecha funcionan en números sin signo; los corrimientos aritméticos trabajan en números con signo. Los corrimientos lógicos multiplican o dividen datos sin signo y los corrimientos aritméticos multiplican o dividen datos con signo. Un corrimiento a la izquierda siempre multiplica por 2 en cada posición de bit desplazada y un desplazamiento a la derecha siempre divide entre 2 en cada posición de bit desplazada.

En la tabla 4-20 se ilustran algunos modos de direccionamiento permitidos para las diversas instrucciones de corrimiento. Hay dos formas diferentes de corrimientos que se ejecutan en cualquier registro (excepto en registros de segmento) o colocación de memoria que va a desplazarse. Una de las formas emplea un contador de corrimiento inmediato; en la otra se emplea el registro CL para mantener el conteo del corrimiento. Se debe tener en cuenta que CL debe tener el conta-

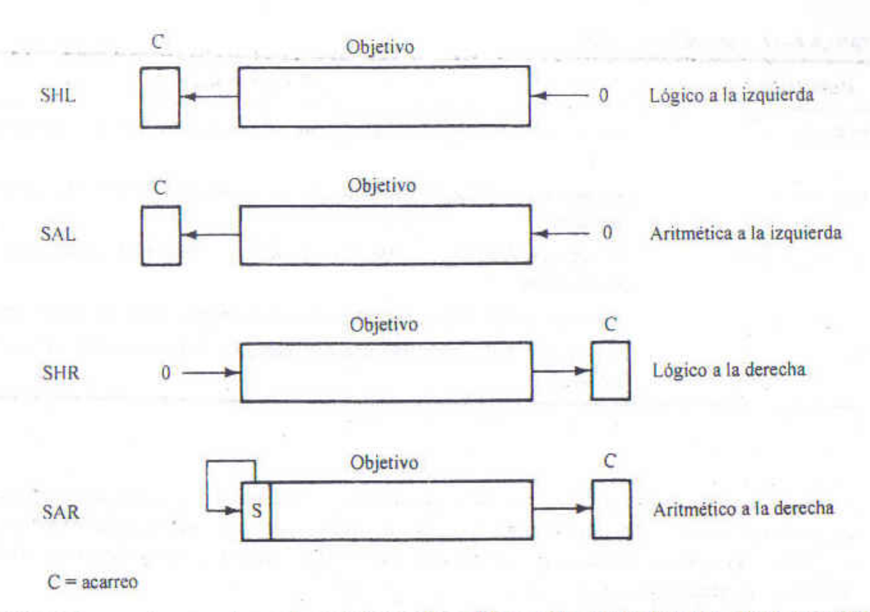


FIGURA 4–9 Las cuatro operaciones de corrimiento disponibles en el conjunto de instrucciones para 8086/8088. Se verá que el objetivo es cualquier registro o localidad de memoria de 8 o de 16 bits, excepto el registro de segmento.

TABLA 4–20 Instrucciones para corrimiento

Instrucción	*Comentario*
SHL AX,1	Hace el corrimiento lógico de AX un lugar a la izquierda
SHR BX,12	Hace el corrimiento lógico de BX 12 lugares a la derecha
SHR ECX,10	Hace el corrimiento lógico de ECX 10 lugares a la derecha
SAL DATA1,CL	Hace el corrimiento aritmético de DATO1, en el segmento de datos, a la izquierda del número de lugares contenido en CL
SAR SI,2	Hace el corrimiento aritmético de SI 2 lugares a la derecha
SAR EDX,14	Hace el corrimiento aritmético de EDX 14 lugares a la derecha

Nota: En los microprocesadores 8086 y 8088 sólo se permite un conteo de 1 de corrimiento inmediato.

dor del corrimiento. Cuando CL es el contador del corrimiento, no cambia cuando se ejecuta la instrucción para desplazamiento. Se debe considerar que el contador del corrimiento es de módulo 32. Esto significa que un conteo de corrimiento de 33, recorrerá los datos un lugar (33/32 = residuo 1).

En el ejemplo 4-28 se muestra la forma en que el registro DX se recorre 14 lugares a la izquierda en dos formas diferentes. En el primer método se utiliza un contador inmediato de 14.

Con el segundo método se carga 14 en CL y, luego, se emplea CL como contador del corrimiento. Ambas instrucciones hacen el corrimiento lógico del contenido del registro DX las posiciones o lugares binarios de 14 bits a la izquierda.

EJEMPLO 4–28

```
0000 C1 E2 0E          SHL  DX,14

                            o

0003 B1 0E             MOV  CL,14
0005 D3 E2             SHL  DX,CL
```

Supóngase que el contenido de AX se debe multiplicar por 10, como se muestra en el ejemplo 4-29. Esto se puede hacer en dos formas: con la instrucción MUL o con corrimientos y sumas. Un número se duplica al desplazarlo a la izquierda un lugar binario. Cuando se duplica un número, entonces al sumarlo al número multiplicado por 8, el resultado es de 10 veces el número. El número 10 decimal es 1010 en binario. Aparece un 1 lógico en las posiciones del 2 y del 8. Si se suman dos veces el número a 8 veces el número, el resultado es 10 veces el número. Con el empleo de esta técnica, se puede escribir un programa para multiplicar por cualquier constante. Esta técnica, es a menudo más rápida que la instrucción MUL.

EJEMPLO 4–29

```
                  ;multiplicar AX por 10 (1010)
                  ;
0000 D1 E0                SHL  AX,1         ;AX multiplicado por 2 .
0002 8B D8                MOV  BX,AX
0004 C1 E0 02             SHL  AX,2         ;AX multiplicado por 8
0007 03 C3                ADD  AX,BX        ;10 × AX
                  ;
                  ;multiplicar AX por 18 (10010)
                  ;
0009 D1 E0                SHL  AX,1         ;AX  multiplicado por 2
000B 8B D8                MOV  BX,AX
000D C1 E0 03             SHL  AX,3         ;AX multiplicado por 16
0010 03 D8                ADD  BX,AX
```

Desplazamientos de doble precisión (Sólo 80386 y 80486). Los 80386 y 80486 contienen dos corrimientos de doble precisión: SHLD (recorrer a la izquierda) y SHRD (recorrer a la derecha). Cada una de estas instrucciones tiene 3 operandos en vez de 2, como en otras instrucciones para corrimiento. Ambas instrucciones funcionan con dos registros de 16 o de 32 bits o con una localidad de memoria y un registro de 16 o de 32 bits.

La instrucción SHRD AX,BX,12 es un ejemplo de una instrucción de doble precisión para corrimiento a la derecha. Esta instrucción hace el corrimiento lógico de AX hacia la derecha en 12 posiciones de bit. Los 12 bits de BX que están más a la derecha se desplazan dentro de los 12 bits de AX que están más a la izquierda. Esta instrucción no cambia el contenido de BX. El contador del corrimiento puede ser inmediato, como en este ejemplo, o se puede encontrar en el registro CL, como en las otras instrucciones para corrimiento.

La instrucción SHLD EBX,ECX,16, recorre a EBX a la izquierda. Los 16 bits de EBX que están más a la izquierda, llenan los 16 bits de EBX que están más hacia la derecha después del

corrimiento. Igual que en casos anteriores, el contenido de ECX, que es el segundo operando, no cambia. Esta instrucción, así como la SHRD, afectan los bits de bandera.

Rotaciones

Las instrucciones para rotación colocan los datos binarios porque hacen «rotar» la información en un registro o localidad de memoria, sea de un extremo a otro o a través de la bandera de acarreo. Se utilizan a menudo para recorrer números con ancho mayor de 16 bits en los microprocesadores 8086-80286, o mayor de 32 bits en los 80386/80486. Las cuatro instrucciones disponibles para rotación aparecen en la figura 4-10.

Los números rotan a través de un registro, una localidad de la memoria y de la bandera C (acarreo) o sólo en un registro y una localidad de memoria. Con cualquier tipo de instrucción para rotación, el programador puede emplear rotación a la derecha o a la izquierda. Los modos de direccionamiento con las rotaciones son los mismos que con los corrimientos. Un contador de rotación puede ser inmediato o estar en el registro CL. En la tabla 4-21 se presentan algunas de las posibles instrucciones para rotación. Si se utiliza CL como contador de rotación, no cambia. Igual que con los corrimientos, el conteo en CL es un conteo de módulo 32.

EJEMPLO 4-30

```
0000 D1 E0          SHL   AX,1
0002 D1 D3          RCL   BX,1
0004 D1 D2          RCL   DX,1
```

Las rotaciones se utilizan, a menudo, para recorrer números anchos a la izquierda o a la derecha. El programa del ejemplo 4-30 recorre el número de 48 bits en los registros DX, BX y AX

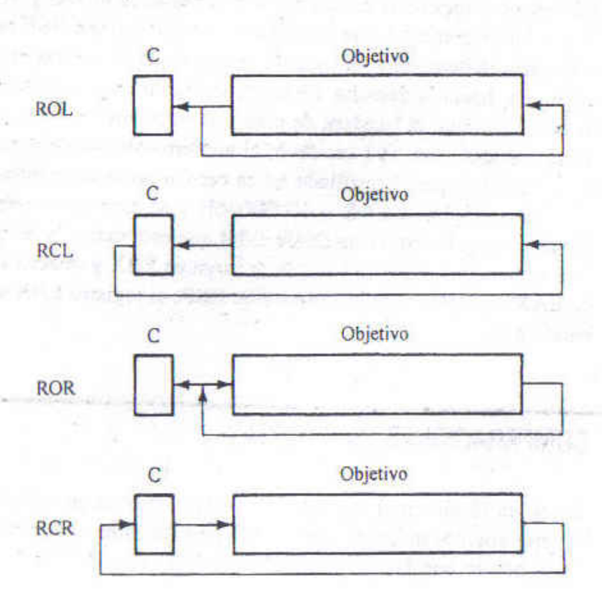


FIGURA 4-10 Las cuatro operaciones de rotación disponibles para el conjunto de instrucciones de 8086/8088. Se verá que las instrucciones RCL y RCR rotan los datos a través del acarreo (C) y que ROL y ROR sólo hacen girar los datos a través del objetivo.

TABLA 4–21 Instrucciones para rotación

Instrucción	*Comentario*
ROL SI,4	SI rota 4 lugares a la izquierda
RCL BL,6	BL rota a la izquierda, 6 lugares
ROL ECX,18	ECX rota 18 lugares a la izquierda
RCR AH,CL	AH rota a la derecha el número de lugares contenido en CL
ROR WORD PTR [BP],2	El contenido palabra de la localidad de memoria en el segmento de pila direccionada por BP rota 2 lugares a la derecha

Nota: En los microprocesadores 8086/8088 sólo se puede emplear un conteo de 1 de rotación inmediata.

un lugar binario hacia la izquierda. Se debe tener en cuenta que el corrimiento de los 16 bits menos significativos (AX) fue el primero. Con esto, se transfiere bit más a la izquierda de AX hacia el bit de bandera C. La siguiente instrucción de rotación hace rotar a C hacia BX y su bit que está más a la izquierda se transfiere a C. La última instrucción hace rotar a C dentro de DX y termina el corrimiento.

Instrucciones para rastreo de bits

Aunque las instrucciones para rastreo de bits no desplazan ni hacen girar un número, rastrean un número en busca de un bit. Debido a que ello se logra dentro del microprocesador con el corrimiento del número, en esta sección se incluyen las instrucciones para rastreo de bits.

Las instrucciones para rastreo de bits BSF (rastrea hacia delante) y BSR (rastrea hacia atrás) sólo están disponibles en los microprocesadores 80386 y 80486. Ambas rastrean un número en busca del primer bit 1 que encuentren. La instrucción BSF rastrea el número desde el bit que está más hacia la derecha y va hacia la izquierda; BSR rastrea el número desde el bit que está más a la izquierda, hacia la derecha. Si se encuentra un bit 1 con cualquiera de las instrucciones de rastreo de bits, se activa la bandera de cero y la posición del bit 1 se carga en el operando de destino. Si no se encuentra un bit 1 (es decir, el número sólo contiene ceros), se desactiva la bandera de cero. Esto significa que el resultado no es cero si no se encuentra ningún bit 1.

Por ejemplo, si EAX = 60000000H y se ejecuta la instrucción BSF, EBX,EAX se rastrea el número hacia la izquierda desde el bit que está más a la derecha. El primer bit 1 que se encuentra está en la posición 29 de bit, que se carga en EBX y se activa la bandera de cero. Si el mismo valor de EAX se utiliza para la instrucción BSR, el registro EBX se carga con un 30 y se activa el bit de bandera cero.

4–6 COMPARACIONES EN CADENAS

Como ya se vio en el capítulo 3, las instrucciones en cadena son muy poderosas porque permiten al programador manejar grandes bloques de datos con relativa facilidad. El manejo de bloques de datos ocurre con las instrucciones para cadenas MOVS, LODS, STOS, INS y OUTS.

En esta sección se describen instrucciones adicionales para cadena que permiten probar una sección de memoria contra una constante o de otra sección de la memoria. Para lograr estas tareas, se utilizan las instrucciones SCAS (*rastrear cadena*) o CMPS (*comparar cadena*).

SCAS

La instrucción para rastrear cadena (SCAS) compara el registro AL con un bloque de bytes de memoria, al registro AX con un bloque de palabras de la memoria, o al registro EAX (sólo 80386 y 80486) con un bloque de dobles palabras de memoria. La instrucción SCAS resta la localidad de memoria de AL, AX o EAX sin alterar ni el registro ni la localidad de la memoria. El código de operación utilizado para la comparación de bytes es SCASB, para la comparación de palabras es SCASW y el utilizado para la comparación de dobles palabras es SCASD. En todos los casos, el contenido de la localidad de memoria es direccionado por DI, en el segmento extra, se compara con AL, AX o EAX. Recuérdese que este segmento implícito no se puede cambiar con un prefijo de cambio de segmento.

En SCAS, igual que en las otras instrucciones para cadena, se emplea la bandera de sentido (D) para seleccionar incremento o decremento automáticos para DI. También repiten si tienen un prefijo *condicional* para repetición.

EJEMPLO 4–31

```
0000 BF 0011 R          MOV   DI,OFFSET BLOQUE   ;direccionar datos
0003 FC                 CLD                      ;autoincremento
0004 B9 0064            MOV   CX,100             ;cargar contador
0007 32 C0              XOR   AL,AL              ;borrar AL
0009 F2/AE              REPNE SCASB              ;buscar
```

Supóngase que hay una sección de memoria de 100 bytes de longitud y que empieza en la localidad BLOQUE. Hay que probar esta sección de la memoria para determinar si en alguna de sus localidades hay un 00H. En el ejemplo 4-31 se muestra la forma de rastrear esta parte de la memoria en busca de un 00H con la instrucción SCASB. En este caso, la instrucción SCASB lleva el prefijo REPNE (*repetir mientras no sea igual*). El prefijo REPNE hace que la instrucción SCASB se repita hasta que, ya sea que el registro CX sea cero o hasta que haya una condición de igualdad como resultado de la comparación con la instrucción SCASB. Otro prefijo condicional para repetir es REPE (*repetir si es igual*). Con cualquiera de los prefijos de repetición, el contenido de CX se decrementa sin alterar los bits de bandera. La instrucción SCASB cambia las banderas.

EJEMPLO 4–32

```
0000              OMITE   PROC  FAR

0000 FC                   CLD                ;autoincremento
0001 B9 0100              MOV   CX,256       ;contador
0004 B0 20                MOV   AL,20H       ;obtener espacio
0006 F3/AE                REPE SCASB         ;buscar
0008 CB                   RET

0009              OMITE   ENDP
```

Supóngase que necesita desarrollar un programa que salte los espacios codificados en ASCII en un arreglo de memoria. Esta tarea se describe en el ejemplo 4-32. En este procedimiento se

supone que el registro DI ya direcciona a la cadena de caracteres en código ASCII y que la longitud de la cadena es de 256 bytes o menos. Debido a que este programa es para saltar espacios (20H), se utiliza el prefijo REPE (repetir si es igual) con la instrucción SCASB, la cual repite la comparación, buscando un 20H, mientras exista la condición de igualdad.

CMPS

La instrucción *comparar cadena* (CPMS) siempre compara dos secciones de datos en la memoria como bytes (CMPSB), palabras (CMPSW) o como dobles palabras (CMPSD). Se debe tener en cuenta que las dobles palabras sólo se pueden usar en los 80386 y 80486. El contenido de la localidad del segmento de datos direccionada por SI se compara con el contenido de la localidad en el segmento extra direccionada por DI. La instrucción CMPS incrementa o decrementa a SI y a DI. La instrucción CMPS, por lo general, se utiliza con un prefijo REPE o uno REPNE. Las alternativas de esos prefijos son REPZ (repetir mientras sea cero) o REPNZ (repetir aunque no sea cero), pero por lo general se utilizan REPE o REPNE.

EJEMPLO 4–33

```
0000              MATCH    PROC  FAR

0000 BE 0075 R             MOV   SI,OFFSET LINEA    ;direccionar LINEA
0003 BF 007F R             MOV   DI,OFFSET TABLA    ;direccionar TABLA
0006 FC                    CLD                      ;incremento automático
0007 B9 000A               MOV   CX,10              ;contador
000A F3/A6                 REPE  CMPSB              ;buscar
000C CB                    RET

000D              MATCH    ENDP
```

En el ejemplo 4-33 se ilustra un procedimiento breve para comparar dos secciones de memoria en busca de igualdad. La instrucción CMPSB lleva el prefijo REPE. Esto hace que la búsqueda continúe mientras haya una condición de igualdad. Cuando el registro CX se vuelve 0, o si hay una condición de desigualdad, la instrucción CMPSB detiene la ejecución. Si CX es cero o las banderas indican una condición igual las dos cadenas concuerdan. Si CX no es cero o las banderas indican una condición de igualdad, las cadenas no concuerdan.

4–7 RESUMEN

1. La suma (ADD) puede ser de 8, 16 o 32 bits. La instrucción ADD permite cualquier modo de direccionamiento excepto el direccionamiento de registro de segmento. Todas las banderas cambian cuando se ejecuta la instrucción ADD. Un tipo diferente de suma, o sea suma con acarreo (ADC), suma dos operandos y el contenido de C. El microprocesador 80486 tiene una instrucción adicional (XADD) que combina la suma con un intercambio.
2. La instrucción (INC) para incrementar, suma un 1 a un byte, palabra o doble palabra de un registro o de datos de la memoria. La instrucción INC afecta todos los bits de bandera, excepto la bandera de acarreo. Los directivos BYTE PTR, WORD PTR y DWORD PTR aparecen

con la instrucción INC cuando se direcciona el contenido de una localidad de memoria con un apuntador.

3. La resta (SUB) es de un byte, palabra o doble palabra y se efectúa en un registro o en una localidad en la memoria. La única forma de direccionamiento no permitida por la instrucción SUB es el direccionamiento a un registro de segmento. La instrucción para restar afecta en todas las banderas y resta el acarreo si se utiliza la forma SBB.
4. La instrucción para decrementar (DEC) resta un uno al contenido de un registro o una localidad en la memoria. Los únicos modos de direccionamiento que no están permitidos con DEC son: direccionamiento inmediato o de registro de segmento. La instrucción DEC no afecta la bandera de acarreo y se utiliza a menudo con BYTE PTR, WORD PTR o DWORD PTR.
5. La instrucción para comparar (CMP) es una forma especial de resta que no almacena la diferencia; en lugar de ello, cambian las banderas para reflejar la diferencia. Esta instrucción se emplea para comparar un byte o palabra completos, ubicados en cualquier registro (excepto de segmento) o en la memoria. Una instrucción (CMPXCHG) para comparar, es una combinación de instrucciones para comparar e intercambiar en el microprocesador 80486.
6. La multiplicación es de byte, palabra o doble palabra y puede ser con signo (IMUL) o sin signo (MUL). La multiplicación de 8 bits siempre multiplica el registro AL por un operando y el producto se encuentra en AX. La multiplicación de 16 bits siempre multiplica el registro AX por un operando y el producto se encuentra en DX-AX. La multiplicación de 32 bits siempre multiplica el registro EAX por un operando y el producto se encuentra en EDX–EAX. Hay una instrucción especial IMUL inmediata en los 80286-80486 que contiene 3 operandos. Por ejemplo, la instrucción IMUL BX,CX,3 multiplica CX por 3 y deja el producto en BX.
7. La división es de byte, palabra o doble palabra y puede ser con signo (IDIV) o sin signo (DIV). Para una división de 8 bits, el registro AX se divide entre el operando, después de lo cual el cociente aparece en AL y el residuo en AH. En la división de 16 bits, el registro DX–AX se divide entre el operando y, luego, el registro AX contiene el cociente y DX el residuo. En la división de 32 bits, el registro EDX–EAX se divide entre el operando, después de lo cual EAX contiene el cociente y el registro EDX el residuo.
8. Los datos en BCD se suman o restan en forma empacada mediante el ajuste del resultado de la suma con DAA o de la resta con DAS. Los datos en ASCII se suman, restan, multiplican o dividen cuando las operaciones se ajustan con AAA, AAS, AAM y AAD.
9. La instrucción AAM tiene una interesante característica adicional, porque le permite convertir un número binario a BCD sin empacar. Esta instrucción convierte un número binario entre 00H y 63H en BCD sin empacar en AX.
10. Las instrucciones AND, OR y OR Exclusivo efectúan funciones lógicas en un byte, palabra o doble palabra almacenados en un registro o en una localidad de memoria. Todas las banderas cambian con estas instrucciones y se hacen cero (borran) acarreo (C) y sobreflujo (O).
11. La instrucción TEST efectúa la operación AND y el producto lógico se pierde. Esta instrucción cambia los bits de bandera para indicar el resultado de la prueba.
12. Las instrucciones NOT y NEG efectúan inversiones lógica y aritmética. La instrucción NOT complementa a 1 un operando y la instrucción NEG complementa a dos un operando.
13. Hay ocho diferentes instrucciones para corrimiento y rotación. Cada una recorre o rota un registro de byte, palabra o doble palabra o datos de la memoria. Estas instrucciones tienen dos operandos: el primero es la de los datos desplazados o girados y, el segundo es un contador de corrimiento inmediato o de rotación o CL. Si el segundo operando es CL, el registro

CL contiene el contador de los corrimientos o rotaciones. En los microprocesadores 80386 y 80486, hay dos instrucciones adicionales para corrimiento, de doble precisión (SHRD y SHLD).

14. La instrucción SCAS para rastrear compara AL, AX o EAX con el contenido de una localidad de memoria en el segmento extra direccionada por DI.
15. La instrucción para comparar cadenas (CMPS) compara el contenido byte, palabra o doble palabra de dos secciones de la memoria. DI direcciona a una sección en el segmento extra y SI a la otra en el segmento de datos.
16. Las instrucciones SCAS y CMPS se repiten con los prefijos REPE o REPNE. El prefijo REPE repite las intenciones para cadena, mientras hay una condición de igualdad y el registro REPNE repite la instrucción a la cadena cuando hay una condición de igualdad.

4–8 CUESTIONARIO Y PROBLEMAS

1. Seleccione una instrucción AND que permita:
   (a) sumar BX con AX
   (b) sumar 12H con AL
   (c) sumar EDI y EBP
   (d) sumar 22H con CX
   (e) sumar los datos direccionados por SI con AL
   (f) sumar CX a los datos almacenados en la localidad RANA en la memoria
2. ¿Qué hay de incorrecto en la instrucción ADD ECX,AX?
3. ¿Es posible sumar CX con DS con la instrucción ADD?
4. Si AX = 1001H y DX = 20FFH enumere la suma y el contenido de cada registro de bandera, después de que se ejecute la instrucción ADD AX,DX.
5. Desarrolle una secuencia corta de instrucciones para sumar AL, BL, CL, DL y AH. Salve la suma en el registro DH.
6. Desarrolle una secuencia corta de instrucciones para sumar AX, BX, CX, DX y SP. Salve la suma en el registro DI.
7. Desarrolle una secuencia corta de instrucciones para sumar ECX, EDX y ESI. Salve la suma en el registro EDI.
8. Seleccione una instrucción para sumar BX con DX y que también sume el contenido de la bandera (C) de acarreo al resultado.
9. Seleccione una instrucción que sume un 1 al contenido del registro SP.
10. ¿Qué hay de incorrecto en la instrucción INC [BX]?
11. Seleccione una instrucción SUB que permita:
    (a) restar BX de CX
    (b) restar OEEH de DH
    (c) restar DI de SI
    (d) restar 3322H de EBP
    (e) restar los datos direccionados por SI, de CH
    (f) restar los datos almacenados 10 palabras después de la localidad direccionada por SI, de DX
    (g) restar AL de la localidad RANA en la memoria

12. Si DL = 0FEH y BH = 72H, muestre la diferencia una vez que se reste BH de DL y muestre el contenido de los bits del registro de bandera.
13. Escriba una secuencia corta de instrucciones para restar los números que hay en DI, SI y BP, del registro AX. Almacene la diferencia en el registro BX.
14. Seleccione una instrucción para restar 1 del registro EBX.
15. Explique lo que se logra con la instrucción SBB [DI - 4],DX.
16. Explique la diferencia entre una instrucción SUB y una CMP.
17. Cuando se multiplican dos números de 8 bits, ¿en qué registro se encontrará el producto?
18. Cuando se multiplican dos números de 16 bits, ¿en cuáles dos registros se encontrará el producto? Indique cuál registro contiene las partes más y menos significativas del producto.
19. Cuando se multiplican dos números, ¿qué ocurre con los bits de bandera O y C?
20. ¿Dónde se almacena el producto de una instrucción MUL EDI?
21. ¿Cuál es la diferencia entre las instrucciones IMUL y MUL?
22. Escriba una secuencia de instrucciones que eleve al cubo el número de 8 bits que se encuentra en DL, en el supuesto de que el contenido inicial de DL sea de 5. Compruebe que su resultado es un número de 16 bits.
23. Describa el funcionamiento de la instrucción IMUL BX,DX,100H.
24. Cuando se dividen dos números de 8 bits, ¿en cuál registro se encuentra el dividendo?
25. Cuando se dividen dos números de 16 bits, ¿en qué registro se encuentra el cociente?
26. ¿Qué tipo de errores se detectan durante la división?
27. Explique la diferencia entre las instrucciones IDIV y DIV.
28. ¿En dónde se encuentra el residuo después de una división de 8 bits?
29. Escriba una secuencia corta de instrucciones para dividir el número que hay en BL entre el número que hay en CL y, luego, multiplique el resultado por 2. La respuesta final debe ser un número de 16 bits almacenado en el registro DX.
30. ¿Qué instrucciones se utilizan con las operaciones aritméticas con BCD?
31. ¿Qué instrucciones se utilizan con las operaciones aritméticas con ASCII?
32. Explique la forma en que la instrucción AAM convierte de binario a BCD.
33. Desarrolle una secuencia de instrucciones para sumar el número BCD de 8 dígitos en AX y BX con el número BCD de 8 dígitos en CX y DX. (AX y CX son los registros más significativos. El resultado después de la suma se debe encontrar en CX y en DX.)
34. Seleccione una instrucción AND que permita:
    (a) BX y DX y salvar el resultado en BX
    (b) 0EAH con DH
    (c) DI con BP y salvar el resultado en DI
    (d) 1122H con EAX
    (e) los datos direccionados por BP con CX y salvar el resultado en la m moria
    (f) los datos almacenados en las cuatro palabras antes de la localidad direccionada por SI, con DX y salvar el resultado en DX
    (g) AL con la localidad de memoria QUE y salvar el resultado en la localidad QUE
35. Desarrolle una secuencia corta de instrucciones que borre los tres bits más a la izquierda en DH, sin cambiar DH y almacene el resultado en BH.
36. Seleccione una instrucción OR que permita:
    (a) ejecutar el OR lógico de BL con AH y salvar el resultado en AH
    (b) ejecutar el OR lógico de 88H con ECX
    (c) ejecutar el OR lógico de DX con SI y salvar el resultado en SI

(d) ejecutar el OR lógico de 1122H con BP
(e) ejecutar el OR lógico de los datos direccionados por BX, con CX y salvar el resultado en la memoria
(f) ejecutar el OR lógico de los datos almacenados 40 bytes después de la localidad direccionada por BP, con AL y salvar el resultado en AL
(g) AH con la localidad CUANDO de la memoria 6 salvar el resultado en CUANDO

37. Desarrolle una secuencia corta de instrucciones que haya uno a los cinco bits más a la derecha de DI, sin cambiar DI. Salve el resultado en SI.
38. Seleccione la instrucción XOR (Or Exclusivo) que permita:
    (a) ejecutar el XOR lógico de BH con AL y salvar el resultado en AH
    (b) ejecutar el XOR lógico de 99H con CL
    (c) ejecutar el XOR lógico de DX con DI y salvar el resultado en DX
    (d) ejecutar el XOR lógico de 1A23H con ESP
    (e) ejecutar el XOR lógico de los datos direccionados por EBX, con DX y salvar el resultado en la memoria
    (f) ejecutar el XOR lógico de los datos almacenados 30 palabras después de la localidad direccionada por BP, con DI y salvar el resultado en DI
    (g) ejecutar el XOR lógico de DI con la localidad de memoria BIEN y salvar el resultado en DI
39. Desarrolle una secuencia de instrucciones que establezca los cuatro bits más a la derecha de AX, borre los tres bits más a la izquierda de AX e invierta los bits 7, 8 y 9 de AX.
40. Describa la diferencia entre AND y TEST.
41. Seleccione una instrucción que pruebe la posición 2 de bit del registro CH.
42. ¿Cuál es la diferencia entre un NOT y un NEG?
43. Seleccione la instrucción correcta para llevar a cabo las siguientes tareas:
    (a) recorrer a DI 3 lugares a la derecha y mover los ceros dentro del bit más a la izquierda
    (b) recorrer a todos los bits de AL un lugar a la izquierda y comprobar que se transfiere un 0 dentro de la posición de bit más a la derecha
    (c) hacer la rotación de todos los bits de AL 3 lugares a la izquierda
    (d) hacer la rotación de la bandera acarreo a la derecha un lugar a través por EDX
    (e) rotar el registro DH un lugar a la derecha y comprobar que el signo del resultado es el mismo que el del número original
44. ¿Qué logra la instrucción SCASB?
45. Para las instrucciones en cadena, DI siempre direcciona los datos que están en el segmento __________.
46. ¿Cuál es la finalidad del bit de bandera D?
47. Explique lo que efectúa el prefijo REPE.
48. ¿Qué condición o condiciones terminarán la instrucción repetida para cadena REPNE SCASB?
49. Describa lo que se logra con la instrucción CMPSB.
50. Desarrolle una secuencia de instrucciones que rastreen una sección de 300H bytes de la memoria llamada LISTA en busca de un 66H.

CAPITULO 5

Instrucciones para control de programas

INTRODUCCION

Las instrucciones para control del programa dirigen su flujo y permiten cambiarlo. A menudo ocurre un cambio en el flujo después que las decisiones tomadas con una instrucción CMP o TEST van seguidas por una instrucción para brinco condicional. En este capítulo se explican las instrucciones para control del programa que incluyen brincos, llamadas, retornos, interrupciones e instrucciones para control de la máquina.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Utilizar instrucciones de brinco condicionales e incondicionales para controlar el flujo de un programa.
2. Utilizar las instrucciones para llamada y retorno a fin de incluir procedimientos en la estructura del programa.
3. Explicar el funcionamiento de las interrupciones y de las instrucciones de control de interrupciones.
4. Utilizar las instrucciones para control de la máquina para modificar los bits de bandera.
5. Utilizar las instrucciones ENTER y LEAVE para entrar y salir de las estructuras de la programación.

5–1 EL GRUPO DE BRINCO

La instrucción para brinco (JMP) es el tipo de instrucciones para control del programa y permite al programador saltar secciones de un programa y transferir el control a cualquier parte de la memoria para la siguiente instrucción. Una instrucción condicional para brinco le permite al

programador tomar decisiones basadas en pruebas numéricas. Los resultados de estas pruebas numéricas se conservan en los bits de bandera, los cuales se prueban después con instrucciones condicionales para brinco.

En esta sección de la obra se abarcan todas las instrucciones para brinco y se ilustra su empleo con programas de muestras. También se ven de nuevo las instrucciones LOOP y LOOP condicional, que se presentaron por primera vez en el capítulo 2.

Brinco incondicional (JMP)

Hay tres tipos de instrucciones para brinco incondicional (véase figura 5-1) disponibles en el conjunto de instrucciones del microprocesador: salto corto, cercano y lejano. El salto corto es una instrucción de 2 bytes que permite transferir el programa a localidades de memoria que están dentro del intervalo de +127 bytes y −128 bytes de la localidad de la memoria después del salto. El brinco cercano, de 3 bytes permite transferir el programa dentro de un intervalo de ±32K bytes (a cualquier lugar) desde la instrucción en el segmento de código actual. El brinco lejano de 5 bytes permite un brinco a cualquier localidad en la memoria dentro de todo el sistema de memoria. A los brincos corto y cercano se les suele llamar *brincos intrasegmento* y a los brincos largos, *brincos intersegmentos.*

En los microprocesadores 80386 y 80486, el brinco corto es en un intervalo de ±2G si la máquina trabaja en el modo protegido y de ±32K bytes si trabaja en el modo real. Para el salto en el modo protegido en 80386 y 80486 se utiliza un desplazamiento de 8- o de 32- bits que no se ilustra en la figura 5-1. El brinco lejano en los 80386 y 80486 permite el brinco a cualquier localidad dentro del intervalo de dirección de 4G bytes de estos microprocesadores.

Brinco corto. A los brincos cortos se les llama *brincos relativos* porque se pueden mover a cualquier lugar en el segmento de código actual sin ningún cambio. Esto se debe a que la direc-

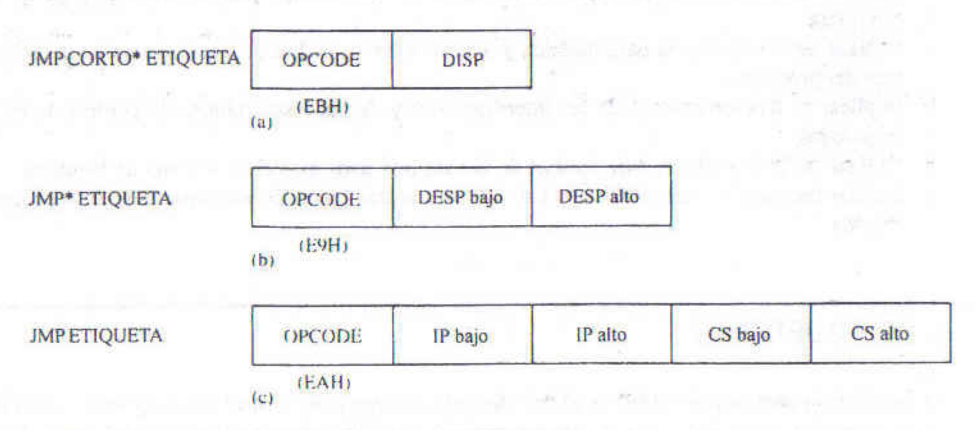


FIGURA 5–1 Los tipos de instrucción para brinco o salto. (a) JMP corto (2 bytes); (b) JMP cercano (3 bytes); (c) JMP lejano (5 bytes).

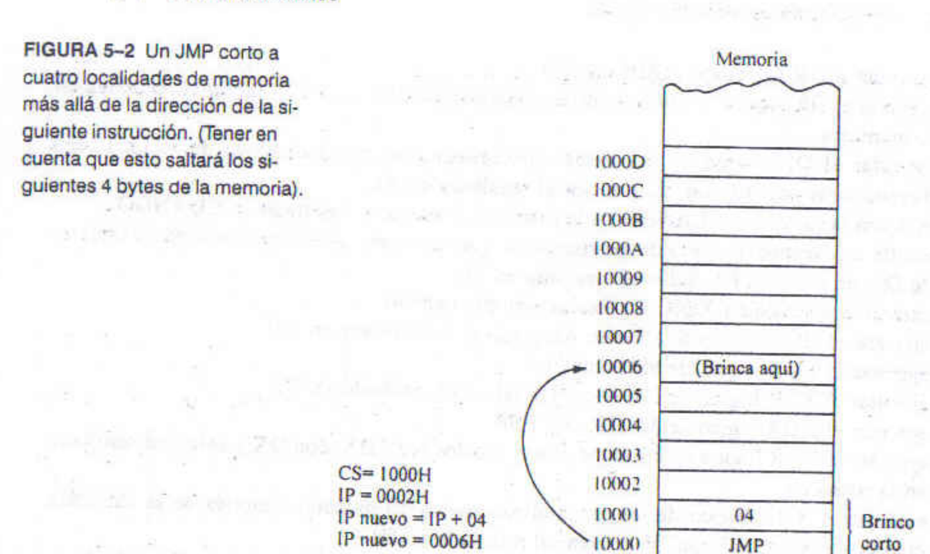


FIGURA 5-2 Un JMP corto a cuatro localidades de memoria más allá de la dirección de la siguiente instrucción. (Tener en cuenta que esto saltará los siguientes 4 bytes de la memoria).

ción para el brinco no se almacena con el código de operación. El código va seguido, en lugar de una dirección para brinco, por una distancia o *desplazamiento*. El desplazamiento en el brinco corto es una *distancia* representada por un número de 1 byte con signo, cuyo valor es entre +127 y −128. La instrucción para el brinco corto aparece en la figura 5-2. Cuando el microprocesador ejecuta un brinco corto, el desplazamiento extiende el signo y suma al apuntador de instrucciones (IP/EIP) para generar la dirección de brinco dentro del segmento de código actual. La instrucción para brinco corto transfiere a esta nueva dirección el programa para la siguiente instrucción.

EJEMPLO 5-1

```
0000 33 DB                  XOR   BX,BX

0002 B8 0001      INICIO:   MOV   AX,1
0005 03 C3                  ADD   AX,BX
0007 EB 17                  JMP   SHORT NEXT

0020 8B D8        SIGUE:    MOV   BX,AX
0022 EB DE                  JMP   START
```

En el ejemplo 5-1 se muestra la forma en que las instrucciones para brinco corto pasan el control de una parte del programa a otra. También se ilustra el empleo de una etiqueta con la instrucción para brinco. Se debe tener en cuenta la forma en que (JMP SHORT NEXT), utiliza el directivo SHORT para obligar a que se produzca un brinco corto, mientras que en la otra no. En casi todos los programas de ensamblador se selecciona la mejor forma de la instrucción para brinco, de modo que la segunda instrucción (JMP INICIO) también se ensambla como un brinco

corto. Si se suma la dirección (0009H) de la siguiente instrucción al desplazamiento extendido por signo (0017H) del primer salto, la dirección de SIGUIENTE es 0017H + 0009H o 0020H.

Siempre que en una instrucción para brinco se menciona una dirección, ésta se determina con una *etiqueta*. Un ejemplo es JMP SIGUE que brinca a la etiqueta SIGUE para la siguiente instrucción. Nunca se utiliza una dirección hexadecimal real con una instrucción para brinco. La etiqueta SIGUE debe ir seguida por dos puntos (SIGUE:) para permitir que la tenga como referencia una instrucción de brinco. Si la etiqueta no va seguida por dos puntos, no se puede saltar a ella. Se debe tener en cuenta que la única vez en que se utilizarán los dos puntos después de una etiqueta, es cuando ésta se emplea con una instrucción para brinco o llamada.

Brinco cercano. El brinco o salto cercano es similar al brinco corto, excepto que su distancia es más larga. Un brinco cercano transfiere el control a una instrucción en el segmento de código actual, ubicado dentro de un intervalo de ±32K bytes de la instrucción para brinco cercano o uno de ±2G en los 80386 y 80486 operando en el modo protegido. El brinco cercano es una instrucción de 3 bytes que contiene un código seguido por un desplazamiento de 16 bits con signo. En los 80386 y 80486, el desplazamiento es de 32 bits y el salto cercano es de 5 bytes de longitud. El desplazamiento con signo se suma al apuntador de instrucciones (IP) a fin de generar la dirección del brinco. Debido a que el desplazamiento con signo está en el intervalo de ±32K bytes, un brinco cercano puede brincar a *cualquier* localidad en la memoria dentro del segmento de código actual en modo real. El segmento de código para modo protegido en los 80386/80486 puede tener una longitud de 4G bytes, por lo cual el desplazamiento de 32 bits permite un brinco cercano a cualquier localidad que esté dentro de ±2G bytes. En la figura 5-3 se ilustra el funcionamiento de una instrucción para brinco cercano en modo real.

El brinco cercano, también se puede cambiar de localidad igual que el brinco corto, porque también es un brinco relativo. Si el segmento de código se mueve a una nueva localidad en la

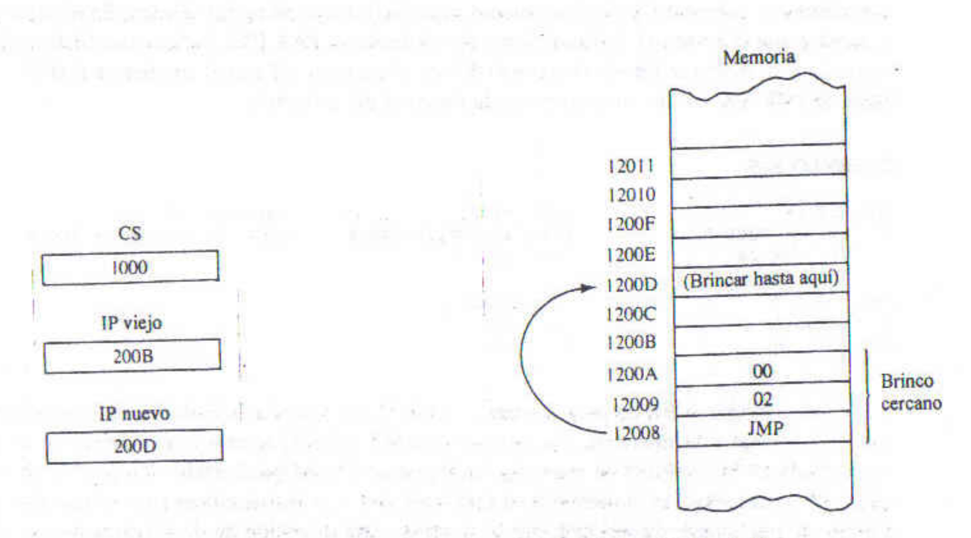


FIGURA 5–3 Un JMP cercano que suma el desplazamiento (0002H) al contenido de IP.

memoria, la distancia entre la instrucción para brinco y la dirección del operando no cambia. Esto permite mover un código de segmento para relocalizarlo. Esta característica, con los segmentos de datos "relocalizables", hace que los microprocesadores de la familia Intel sean ideales para emplearse en un sistema de computadora de uso general. La programación se puede escribir y cargar en cualquier lugar en la memoria y funciona sin modificación, debido a que los brincos relativos y a que los segmentos de datos se pueden relocalizar. Esto sólo ocurre en muy pocos microprocesadores de otras marcas.

EJEMPLO 5–2

```
0000 33 DB                  XOR  BX,BX

0002 B8 0001     INICIO:    MOV  AX,1
0005 03 C3                  ADD  AX,BX
0007 E9 0200 R              JMP  NEXT

0200 8B D8       SIGUE:     MOV  BX,AX
0202 E9 0002 R              JMP  START
```

En el ejemplo 5-2 se muestran los mismos programas básicos que en el ejemplo 5-1, excepto que la distancia para el brinco es mayor. El primer brinco (JMP SIGUE) transfiere el control a la instrucción en la localidad 0200H de la memoria dentro del segmento de código. Se debe tener en cuenta que la instrucción se ensambla como una E9 0200 R. La letra R denota la dirección "relocalizable" del brinco de 0200H. La instrucción real en lenguaje de máquina se ensambla como E9 F6 01 que no aparece en el listado del ensamblador. El desplazamiento real para esta instrucción de brinco es un 01F6H. En el listado del ensamblador aparece que la dirección actual para el salto es 0200 R, por lo cual es más fácil interpretar la dirección mientras se desarrolla el programa. Si se pudiera ver el archivo del ligado hexadecimal, se vería que ese brinco se ensambló como E9 F6 01.

Brinco lejano. Los brincos lejanos (figura 5-4) adquieren nuevas direcciones de segmento y desplazamiento para lograr el brinco. Los bits 2 y 3 de esta instrucción de 5 bytes contienen la nueva dirección del desplazamiento y los bytes 4 y 5 contienen la nueva dirección del segmento. Si se hace funcionar un microprocesador 80286-80486 en el modo protegido, la dirección del segmento accesa a un descriptor, el cual contiene la dirección base del segmento para salto lejano; la dirección de desplazamiento, que es de 16 o de 32 bits contiene la localidad del desplazamiento dentro del nuevo segmento de código.

EJEMPLO 5–3

```
                            EXTRN  ARRIBA:FAR

0000 33 DB                  XOR    BX,BX

0002 B8 0001     INICIO:    MOV    AX,1
0005 03 C3                  ADD    AX,BX
0007 E9 0200 R              JMP    NEXT

0200 8B D8       SIGUE:     MOV    BX,AX
0202 EA 0002--R             JMP    FAR PTR INICIO

0207 EA 0000--E             JMP    ARRIBA
```

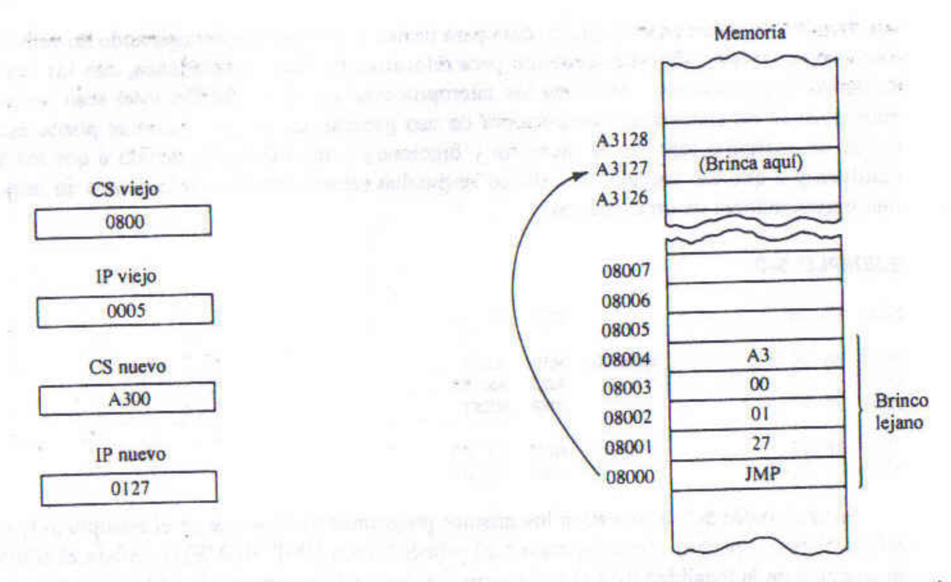


FIGURA 5–4 Un JMP lejano que sustituye al contenido de los registros CS e IP con los 4 bytes que siguen al código de operación.

En el ejemplo 5-3 se presenta un programa corto en el que se emplea una instrucción para brinco lejano; ésta aparece a veces, con el directivo FAR PTR tal como se ilustra. Otra forma de obtener un brinco lejano es definir a una etiqueta como etiqueta lejana; ésta sólo es lejana si está en el exterior del segmento de código en curso. La instrucción JMP ARRI en el ejemplo consulta una etiqueta lejana. La etiqueta ARRI la define como etiqueta lejana con el directivo EXTRN ARRI:FAR. Las etiquetas externas sólo aparecen en programas que contienen más de un archivo de programa.

Cuando se unen los archivos del programa, el ligador introduce la dirección de la etiqueta ARRI en la instrucción JMP ARRI. También introduce la dirección del segmento en la instrucción JMP INICIO. La dirección del segmento en JMP FAR INICIO aparece en el listado como —*R* para indicar que es "relocalizable" o como —*E* para indicar que es externa. En ambos casos, la raya (—) que precede a esas letras la agrega el ligador cuando liga o une los archivos del programa.

Brincos con operandos de los registros. En las instrucciones para brinco se puede especificar, como operando, un registro de 16 o de 32 bits. Esto establece a la instrucción en forma automática, como brinco indirecto. La dirección del brinco está en el registro especificado por la instrucción. Al contrario del desplazamiento relacionado con el brinco cercano, el contenido del registro se transfiere directamente al apuntador de instrucciones (IP); no se suma al apuntador como ocurre en los brincos corto y cercano. La instrucción JMP AX, por ejemplo, copia el contenido del registro AX y lo pone en el IP cuando ocurre el salto. Esto permite un brinco a cualquier localidad dentro del segmento de código actual. En los 80386 y 80486, la instrucción JMP EAX también salta a cualquier localidad dentro del segmento de código actual; la diferencia está en

que en el modo protegido, el segmento de código puede tener una longitud de 4G bytes, por lo cual se necesita una dirección de desplazamiento de 32 bits.

EJEMPLO 5–4

```
0000 03 F6                    ADD  SI,SI             ;duplicar SI
0002 81 C6 000B R             ADD  SI,OFFSET TABLA   ;sumar dirección de TABLA
0006 2E: 8B 04                MOV  AX,CS:[SI]        ;obtener dirección para brinco
0009 FF E0                    JMP  AX

000B 1000 R         TABLE:    DW   CERO
000D 2000 R                   DW   UNO
000F 3000 R                   DW   DOS
```

En el ejemplo 5-4 se muestra la forma en que la instrucción JMP AX accesa una tabla de brincos. Para esta secuencia de instrucciones se supone que SI contiene un 0, 1 o 2. Debido a que la tabla de brincos contiene direcciones de desplazamiento de 16 bits, el contenido de SI se duplica a 0, 2 o 4, por lo cual se puede accesar a una entrada de 16 bits en la tabla. Luego, la dirección de desplazamiento del inicio de la tabla se suma a SI para consultar una dirección de brinco. La instrucción MOV AX,CS:[SI] extrae una dirección de la tabla de brincos del segmento de código a fin de que la instrucción JMP AX brinque a la localidad indicada en la tabla.

Con este ejemplo se muestra la forma de "extraer" la dirección CERO de la tabla si aparece un cero en SI. Una vez que se tiene la dirección, la instrucción JMP AX hace que el programa se transfiera a la dirección CERO de la tabla. Lo mismo ocurre para un valor inicial de 1 o de 2 en el registro SI. Estos números accesan a las direcciones UNO y DOS de las localidades de brinco de la tabla.

Brinco indirecto con el empleo de un índice. En la instrucción para brinco también se puede emplear la forma entre corchetes [ ] de direccionamiento para acceder en forma directa a la tabla de brincos; ésta puede contener direcciones de desplazamiento para brincos indirectos cercanos o direcciones de segmento, y desplazamiento para los brincos indirectos lejanos. En el ensamblador se supone que el brinco es cercano, salvo que el directivo FAR PTR indique una instrucción para brinco lejano. Ahora se repetir el ejemplo 5-4 en el ejemplo 5-5 con el empleo de JMP CS:[SI] en lugar de JMP AX, con lo cual se reduce la longitud del programa.

EJEMPLO 5–5

```
0000 03 F6                    ADD  SI,SI             ;duplicar SI
0002 81 C6 0009 R             ADD  SI,OFFSET TABLA   ;sumar dirección de TABLA
0006 2E: FF 24                JMP  CS:[SI]

0009 1000 R         TABLA:    DW   CERO
000B 2000 R                   DW   UNO
000D 3000 R                   DW   DOS
```

El mecanismo utilizado para accesar a la tabla de brincos es idéntico al de una referencia o consulta normal a la memoria. La instrucción JMP CS:[SI] apunta a una dirección de brinco almacenada en la localidad de memoria en segmento de código direccionada por SI. Brinca a la dirección almacenada en la memoria en esta localidad. Las instrucciones para brinco por registro e indirecto por índice, direccionan por lo general a una dirección de desplazamiento de 16 bits.

Esto significa que ambos tipos de brincos son cercanos. Si se ejecuta una instrucción JMP FA PTR [SI] se supone en el microprocesador que la tabla para brinco contiene direcciones (IP y CS) de doble palabra de 32 bits.

Brincos y activamientos condicionales

Los brincos condicionales son siempre *cortos* en los microprocesadores 8086-80286. Esto limita el alcance del brinco a entre +127 bytes y -128 bytes desde la localidad que sigue del salto condicional. En los 80386 y 80486 los brincos condicionales son cortos o cercanos. Esto permite un brinco condicional a cualquier localidad dentro del segmento de código actual en los 80386 y 80486. En la tabla 5-1 se presentan todas las instrucciones para brinco condicional y sus condiciones para la prueba.

Las instrucciones para brinco condicional prueban los siguientes bits de bandera: signo (S), cero (Z), acarreo (C), paridad (P) y sobreflujo (O). Si la condición que se prueba es verdadera, ocurre una transferencia a la etiqueta relacionada con la instrucción de brinco. Si la condición es falsa, se ejecuta la siguiente instrucción en la secuencia del programa.

El funcionamiento de la mayor parte de las instrucciones para salto condicional es directo, porque a menudo prueban un solo bit de bandera, pero algunas prueban más de una bandera. Las

TABLA 5–1 Instrucciones para brinco (salto) condicional

Instrucción	*Condición probada*	*Comentario*
JA	C = 0 y Z = 0	Brinca si está por arriba
JAE	C = 0	Brinca si está por arriba o es igual a
JB	C = 1	Brinca si está por abajo
JBE	C = 1 o Z = 1	Brinca si está por abajo o es igual a
JC	C = 1	Brinca por acarreo
JE o JZ	Z = 1	Brinca si es igual o cero
JG	Z = 0 y S = O	Brinca si es mayor que
JGE	S = O	Brinca si es mayor que, o igual a
JL	S ≠ O	Brinca si es menor que
JLE	Z = 1 o S ≠ O	Brinca si es menor que, o igual a
JNC	C = 0	Brinca si no hay acarreo
JNE o JNZ	Z = 0	Brinca si no es igual a, o si no es cero
JNO	O = 0	Brinca si no hay sobreflujo
JNS	S = 0	Brinca si no hay signo
JNP/JPO	P = 0	Brinca si no hay paridad o con paridad impar
JO	O = 1	Brinca si hay sobreflujo
JP/JPE	P = 1	Brinca si hay paridad o paridad par
JS	S = 1	Brinco si hay signo
JCXZ	CX = 0	Brincar si CX = 0
JECXZ	ECX = 0	Brincar si ECX = 0 (Sólo 80386 y 8-0486)

encuentra en la tabla. Otro método utilizado para ver si se encuentran los datos es la instrucción JNE; si ésta sustituye a JCXZ, desempeña la misma función. Después de que se ejecuta la instrucción SCASB, las banderas indicarán una condición de no es igual, si no se encontraron los datos en la tabla.

EJEMPLO 5–6

```
                        ;procedimiento que busca en una tabla de 100 bytes
                        ;para ver si hay un OAH
                        ;
0000                    SCAN      PROC   NEAR

0000 BE 0000 R                    MOV    SI,OFFSET TABLA    ;direccionar a TABLA
0003 B9 0064                      MOV    CX,100             ;cargar conteo
0006 B0 0A                        MOV    AL,0AH             ;cargar datos de la búsqueda
0008 FC                           CLD
0009 F2/AE                        REPNE SCASB
000B E3 23                        JCXZ   NO_ESTA
000D C3                           RET

000E                    SCAN      ENDP
```

Instrucciones condicionales para activar. Además de las instrucciones condicionales para brinco, los microprocesadores 80386 y 80486 también incluyen instrucciones condicionales para activar. Las condiciones probadas por los brincos condicionales se ponen a trabajar con las instrucciones condicionales para "activar" que hacen un byte 1 o 0, según sea la condición que se prueba. En la tabla 5-2 se presentan las formas disponibles de instrucciones condicionales para activar en los 80386 y 80486.

Estas instrucciones son útiles cuando se debe probar una condición en un momento o punto posterior en el programa. Por ejemplo, se puede activar un byte para indicar que el acarreo se desactivó en un punto en el programa con la instrucción SETNC MEM. Esta instrucción carga un 01H en la localidad MEM en la memoria si no hubo acarreo y carga un 00H en MEM si se activó el acarreo. El contenido de MEM se puede probar en cualquier punto posterior en el programa, para determinar si se desactivó el acarreo en el punto en donde se ejecutó la instrucción SETNC MEM.

LOOP

La instrucción LOOP es una combinación de decremento CX y de un brinco condicional. En los 8086-80286, LOOP decrementa a CX y si éste no es igual a cero, brinca a la dirección indicada por la etiqueta. Si CX es cero se ejecuta la siguiente instrucción. En los 80386 y 80486, LOOP decrementa a CX o a ECX según sea el modo de la instrucción. Si los 80386 y 80486 funcionan en el modo de instrucción de 16 bits, LOOP utiliza a CX; si funcionan en el modo de instrucción de 32 bits, LOOP utiliza a ECX. Esto se puede cambiar con LOOPW (con el empleo de CX) y de LOOPD (con el empleo de ECX), sólo en los 80386 y 80486.

En el ejemplo 5-7 se muestra la forma en que los datos en un bloque de memoria (BLOCK1) se suman a los datos en un segundo bloque de memoria (BLOCK2) con el empleo de LOOP para controlar cuántos números se van a sumar. Las instrucciones LODSW y STOSW accesan a los datos en BLOCK1 y en BLOCK2. La instrucción ADD AX,ES:[DI] accesa a los datos en BLOCK2

FIGURA 5-5 Los números con signo y sin signo siguen diferentes órdenes.

Números sin signo

255	FFH
254	FEH
132	84H
131	83H
130	82H
129	81H
128	80H
4	04H
3	03H
2	02H
1	01H
0	00H

Números con signo

+127	7FH
+126	7EH
+2	02H
+1	01H
+0	00H
−1	FFH
−2	FEH
+124	84H
+125	83H
+126	82H
−127	81H
−128	80H

comparaciones de magnitud relativa requieren instrucciones más complicadas para el brinco condicional y deben probar más de un bit de bandera.

Debido a que se utilizan números con signo y sin signo y a que el orden de estos números es diferente, hay dos conjuntos de instrucciones para salto condicional con comparación de magnitud. En la figura 5-5 se muestra el orden de los números de 8 bits con signo y sin signo. Para los números de 16 y de 32 bits se siguió el mismo orden que para los de 8 bits, excepto que son más grandes. Se verá que hay un FFH por arriba del 00H en el grupo de números sin signo, pero un FFH (−1) es menor que 00H en números con signo. Por tanto, un FFH sin signo está *por arriba* de 00H, pero un FFH con signo es *menor que* 00H.

Cuando se comparan números con signo se emplean JG, JE, JGE, JLE, JE y JNE. Los términos mayor que y menor que se aplican a los números con signo. Cuando se comparan números sin signo, se utilizan JS, JB, JAE, JBE y JNE. Los términos por arriba y por abajo se aplican a números sin signo.

Los siguientes brincos condicionales prueban bits de bandera individuales como sobreflujo y paridad. Se verá que JE tiene un código JZ alterno. Todas las instrucciones tienen alternos, pero muchos de ellos no se emplean en programación, porque no hay suficientes razones para ello. (Los códigos alternos aparecen en el apéndice B en el listado del conjunto de instrucciones.) Por ejemplo, la instrucción JA (brinca si está por arriba) tiene a JNBE (brinca si no está por abajo o es igual). Una instrucción JA actúa exactamente igual que una JNBE, pero ésta parece ser inconveniente al compararla con JA.

Las instrucciones más radicales para brinco condicional son JCXZ (brinca si CX =0) y JECXZ (brinca si ECX = 0). Estas son las únicas instrucciones para brinco condicional que no prueban los bits de bandera. En lugar de ello, JCXZ hace una prueba directa del contenido del registro CX sin afectar los bits de bandera y JECXZ prueba el contenido del registro ECX. Si CX =0, ocurre el brinco y si CX no =0 es igual a 0, no ocurre brinco. Asimismo, para la instrucción JECXZ, si ECX = 0 ocurre un brinco y si CX no =0 es igual a cero, no ocurre brinco.

En el ejemplo 5-6 se presenta un programa que utiliza JCXZ. En este caso, la instrucción SCASB busca en una tabla para ver si hay un byte 0AH. Después de la búsqueda, una instrucción JCXZ prueba si el conteo en CX se ha vuelto cero. Si ese conteo es cero, el 0AH no se

TABLA 5–2 Instrucciones para activamiento condicional

Instrucción	*Condición probada*	*Comentario*
SETB	C = 1	Establecer byte si es hacia abajo
SETAE	C = 0	Establecer byte si es hacia arriba o igual
SETBE	C = 1 o Z = 1	Establecer byte si es hacia abajo o igual
SETA	C = 0 y Z = 0	Establecer byte si es hacia arriba
SETE/SETZ	Z = 1	Establecer byte si es igual; establece byte si es cero
SETNE/SETNZ	Z = 0	Establecer byte si no es igual; establecer byte si no es cero
SETL	S ≠ O	Establecer byte si es menor que
SETLE	Z = 1 o S ≠ O	Establecer byte si es menor que, o igual
SETG	Z = 0 y S = O	Establecer byte si es mayor que
SETGE	S = O	Establecer byte si es mayor que, o igual
SETS	S = 1	Establecer byte si hay signo (negativo)
SETNS	S = 0	Establecer byte si no hay signo (positivo)
SETC	C = 1	Establecer byte si hay acarreo
SETNC	C = 0	Establecer byte si no hay acarreo
SETO	O = 1	Establecer byte si hay sobreflujo
SETNO	O = 0	Establecer byte si no hay sobreflujo
SETP	P = 1	Establecer byte si hay paridad par
SETNP	P = 0	Establecer byte si hay paridad impar

ubicados en el segmento extra. La única razón por la cual BLOCK2 está en el segmento es que DI direcciona a los datos del segmento extra con la instrucción STOSW.

EJEMPLO 5–7

```
                    ;procedimiento que agrega una palabra en BLOCK1 y BLOCK2
                    ;
0000                AGREGA    PROC NEAR

0000 B9 0064                  MOV  CX,100              ;cargar conteo
0003 BE 0064 R                MOV  SI,OFFSET BLOCK1    ;direccionar BLOCK1
0006 BF 0000 R                MOV  DI,OFFSET BLOCK2    ;direccionar BLOCK2
0009                DE NUEVO:
0009 AD                       LODSW                    ;get BLOCK1 data
000A 26: 03 05                ADD  AX,ES:[DI]          ;add BLOCK2 data
000D AB                       STOSW                    ;store in BLOCK2
000E E2 F9                    LOOP AGAIN               ;repeat 100 times
0010 C3                       RET

0011                ADDS      ENDP
```

LOOP condicionales. Igual que REP, la instrucción LOOP también tiene formas condicionales: LOOPE y LOOPNE. La instrucción LOOPE (repite mientras sea igual), retiene un ciclo de programación si CX es menor de 0, mientras existe una condición de igualdad. Saldrá del ciclo si

la condición no es igual o si el registro CX se decrementa a 0. La instrucción LOOPNE (repite aunque no sea igual) repite un ciclo de programación si CX no es igual a 0, mientras haya una condición de no igualdad. Saldrá del ciclo si la condición es igual o si el registro CX se decrementa a 0. En los 80386 y 80486, la instrucción LOOP condicional puede utilizar a CX o a ECX como contador. Las instrucciones LOOPEW, LOOPED, LOOPNEW o LOOPNED invalidan o cambian el modo de instrucción si es necesario.

Igual que para las instrucciones condicionales para repetición, LOOPE y LOOPNE tienen sus alternas. La instrucción LOOPE es la misma que LOOPZ y la LOOPNE es la misma que LOOPNZ. En casi todos los programas sólo se aplican LOOPE y LOOPNE.

5–2 PROCEDIMIENTOS

El procedimiento o *subrutina* es parte importante de la arquitectura de cualquier sistema de computadora. Un **procedimiento** es un grupo de instrucciones que, por lo general, desempeñan una tarea. Un procedimiento es una sección de un programa que se puede volver a utilizar y que se almacena una vez en la memoria, pero se emplea tan a menudo como se necesite. Esto ahorra espacio en la memoria y facilita el desarrollo de la programación. La única desventaja de un procedimiento es que la computadora requiere un poco de tiempo para ligarse con el procedimiento y retornar desde él. La instrucción CALL liga el procedimiento y la instrucción RET retorna del procedimiento.

La dirección para el retorno se almacena en la pila siempre que se llama a un procedimiento durante la ejecución de un programa. La instrucción CALL salva la dirección de la instrucción, que va después de ella, en la pila. La instrucción RET recupera una dirección de la pila a fin de que el programa retorne a la instrucción que sigue al CALL.

EJEMPLO 5–8

```
0000                SUMS    PROC   NEAR

0000 03 C3                  ADD    AX,BX
0002 03 C1                  ADD    AX,CX
0004 03 C2                  ADD    AX,DX
0006 C3                     RET

0007                SUMS    ENDP

0007                SUMS1   PROC   FAR

0007 03 C3                  ADD    AX,BX
0009 03 C1                  ADD    AX,CX
000B 03 C2                  ADD    AX,DX
000D CB                     RET

000E                SUMS1   ENDP
```

Hay algunas reglas finitas en el ensamblador para el almacenaje de procedimientos. Un procedimiento empieza con el directivo PROC y termina con el directivo ENDP. Cada directivo aparece con el nombre del procedimiento. Esta estructura facilita localizar el procedimiento en

un listado del programa. El directivo PROC va seguido por el tipo de procedimiento NEAR o FAR. En el ejemplo 5-8 se ilustra la forma en que el ensamblador requiere la definición de un procedimiento cercano (intrasegmento) y de uno lejano (intersegmentos). En la versión 6.0 del MASM, el tipo NEAR o FAR puede ir seguido por el enunciado USES; éste permite, en forma automática, que se pueda salvar en la pila y recuperar de ella cualquier número de registros dentro del procedimiento.

Cuando se comparan estos dos procedimientos, la única diferencia es el código de operación de la instrucción para retorno. En la instrucción para retorno cercana, se emplea el código C3H; en la lejana, el CBH. Un retorno cercano recupera un número de 16 bits de la pila y lo carga en el apuntador de instrucciones para retornar del procedimiento, en el segmento de código actual. Un retorno lejano recupera un número de 32 bits de la pila y lo carga en IP y en CS para retornar desde el procedimiento a cualquier localidad en la memoria.

La mayor parte de los procedimientos que se van a utilizar con todos los programas (*globales*) se deben redactar como procedimientos lejanos. Los procedimientos que se utilizan para una tarea dada (*locales*) se suelen definir como procedimientos cercanos.

CALL

La instrucción CALL transfiere el flujo del programa al procedimiento. La instrucción CALL difiere de las instrucciones para brinco porque CALL salva una dirección para retorno en la pila. La dirección de retorno devuelve el control a la instrucción que sigue a CALL en un programa, cuando se ejecuta una instrucción RET.

CALL cercana. La instrucción CALL cercana tiene 3 bytes de longitud; el primer byte contiene el código y el segundo y tercero contienen el desplazamiento o distancia de ±32K en los 8086-80286. La forma de instrucción es idéntica a la de brinco cercano. En los 80386 y 80486 se emplea un desplazamiento de 32 bits cuando trabajan en el modo protegido a fin de permitir una distancia de ±2G bytes. Cuando se ejecuta un CALL cercano, primero salva en la pila la dirección de desplazamiento de la siguiente instrucción. Esa dirección está en el apuntador de instrucciones (IP o EIP). Después de salvar la dirección de retorno, suma los desplazamientos de los bytes 2 y 3 a IP para transferir el control al procedimiento. No hay instrucción CALL corta.

¿Por qué salvar IP o EIP en la pila? El apuntador de instrucciones siempre apunta a la siguiente instrucción en el programa. Para la instrucción CALL, se salva el contenido de IP o EIP dentro de la pila, con lo cual el control del programa pasa a la instrucción que sigue al CALL después de que termina un procedimiento. En la figura 5-6 se ilustran la dirección de retorno (IP) salva en la pila y la llamada procedimiento para los 8086-80286.

CALL lejana. La instrucción CALL lejana es como un brinco lejano porque puede llamar a un procedimiento almacenado en cualquier localidad de la memoria en el sistema. La instrucción CALL lejana tiene 5 bytes y tiene un código de operación seguido por el valor de los registros IP y CS. Los bytes 2 y 3 contienen el nuevo contenido de IP y los bytes 4 y 5 contienen el nuevo contenido de CS.

La instrucción CALL lejana salva el contenido de IP y de CS en la pila antes de brincar a la dirección indicada por los bytes 2 a 5 de la instrucción. Esto permite que la CALL lejana llame a un procedimiento ubicado en cualquier lugar en la memoria y que retorne desde el procedimiento.

FIGURA 5–6 El efecto de una instrucción CALL cercana en la pila y en los registros SP, SS e IP. Se verá la forma en que se almacena IP viejo en la pila.

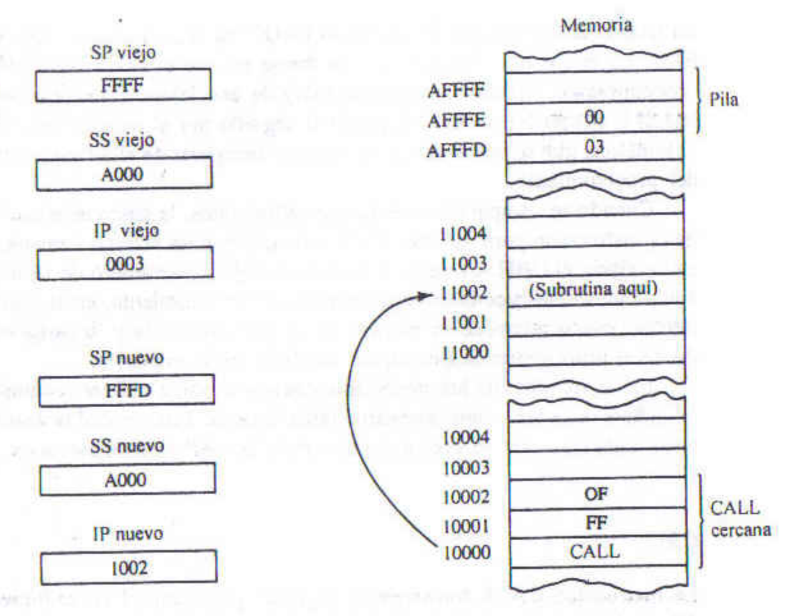
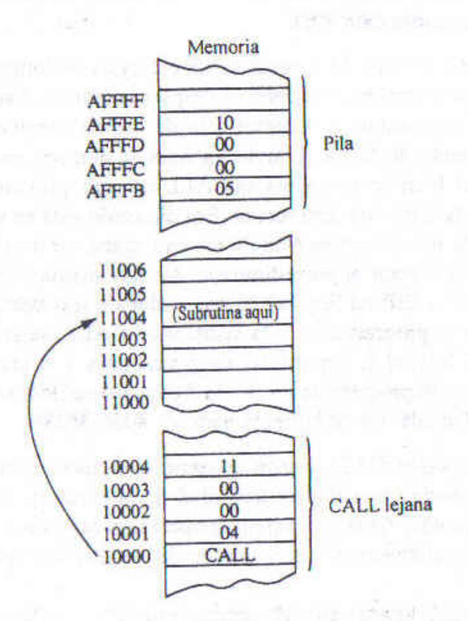


FIGURA 5–7 La instrucción CALL lejana.



En la figura 5-7 se ilustra la forma en que la instrucción CALL lejana llama a un procedimiento lejano. En este caso, se salva el contenido de IP y CS dentro de la pila. Luego, el programa se transfiere al procedimiento.

Instrucciones CALL con operandos registro. Las instrucciones CALL al igual que las JMP pueden contener un operando registro y un ejemplo es la instrucción CALL BX. Esta instrucción salva el contenido de IP dentro de la pila. Después, brinca a la dirección de desplazamiento, ubicada en el registro BX en el segmento de código actual. Este tipo de llamada siempre utiliza una dirección de desplazamiento de 16 bits almacenada en cualquier registro de 16 bits, excepto los registros de segmentos.

EJEMPLO 5-9

```
                    ;llamada a la secuencia
                    ;
0000 BE 0005 R                  MOV  SI,OFFSET COMP
0003 FF D6                      CALL SI

                    ;
                    ;procedimiento COMP
                    ;
0005                COMP        PROC NEAR

0005 52                         PUSH DX
0006 BA 03F8                    MOV  DX,03F8H
0009 EC                         IN   AL,DX
000A 42                         INC  DX
000B EE                         OUT  DX,AL
000C 5A                         POP  DX
000D C3                         RET

000E                COMP        ENDP
```

En el ejemplo 5-9 se ilustra el empleo de la instrucción CALL con un registro para llamar un procedimiento que comienza en la dirección de desplazamiento COMP, la cual se carga al registro SI y, luego, la instrucción CALL SI llama al procedimiento que empieza en la dirección COMP.

Instrucciones CALL a direcciones indirectas en la memoria. La instrucción CALL a una dirección indirecta en la memoria es de particular utilidad siempre que se desea escoger diferentes subrutinas en un programa. Este proceso de selección va ligado frecuentemente con un número que direcciona a una dirección de llamada en una tabla de consulta.

EJEMPLO 5-10

```
                    ;tabla de consulta
                    ;
0000 0100 R         TABLA       DW    UNO
0002 0200 R                     DW    DOS
0004 0300 R                     DW    TRES
                    ;
                    ;llamada a la secuencia
                    ;
0006 4B                         DEC   BX              ;escalar BX
0007 03 DB                      ADD   BX,BX           ;duplicar BX
```

```
0009 BF 0000 R              MOV   DI,OFFSET TABLE   ;direccionar TABLA
000C 2E: FF 11              CALL  CS:[BX+DI]
                 ;
                 ;procedimiento
                 ;
0100             UNO        PROC  NEAR
                 .          .
                 .          .
0100             ONE        ENDP
0200             DOS        PROC  NEAR
                 .          .
                 .          .
0200             TWO        ENDP
0300             TRES       PROC  NEAR
                 .          .
                 .          .
0300             THREE      ENDP
```

En el ejemplo 5-10 se ilustran tres subrutinas separadas indicadas con los números 1, 2 y 3 en AL. La secuencia para llamar ajusta el valor de AL y lo extiende a un número de 16 bits antes de sumarlo al desplazamiento de la tabla de consulta. Esto sirve como referencia para una de las tres subrutinas con el empleo de la instrucción CALL CS:[BX + DI]. El prefijo CS aparece antes del operando de la instrucción CALL porque la TABLA está en el segmento de código en este ejemplo.

La instrucción CALL también puede referenciar apuntadores lejanos si la instrucción aparece como CALL FAR PTR [SI]. Esta instrucción recupera una dirección de 32 bits en la memoria del segmento de datos direccionada por SI y la utiliza como dirección para un procedimiento lejano.

RET

La instrucción para retorno (RET) extrae un número de 16 bits (retorno cercano) de la pila y lo coloca en IP o bien un número de 32 bits (retorno lejano) y lo coloca en IP y CS. Las instrucciones para retorno cercano y lejano están definidas en el directivo PROC para el procedimiento. Esto selecciona en forma automática la instrucción adecuada para el retorno. En los 80386 y 80486 que trabajen en el modo protegido, el retorno lejano extrae 6 bytes de la pila; los primeros cuatro contienen el nuevo valor para EIP y los dos últimos el nuevo valor para CS. El retorno cercano para los 80386 y 80486 en modo protegido, extrae 4 bytes de la pila y los coloca en EIP.

Cuando se cambian IP o EIP o IP o EIP y CS, la dirección de la siguiente instrucción está en una nueva localidad en la memoria, la cual es la dirección de la instrucción que va inmediatamente después de la llamada más reciente a un procedimiento. En la figura 5-8 se ilustra la forma en que una instrucción CALL se liga con un procedimiento y la forma en que la instrucción RET retorna en los microprocesadores 8086-80286.

Hay otra forma más para la instrucción de retorno, la cual suma un número al contenido del apuntador de pila (SP) antes del retorno. Si hay que borrar lo salvado en la pila antes de un retorno, el desplazamiento de 16 bits se suma a SP antes de que el retorno recupere la dirección de retorno en la pila. El efecto es omitir o borrar datos de la pila.

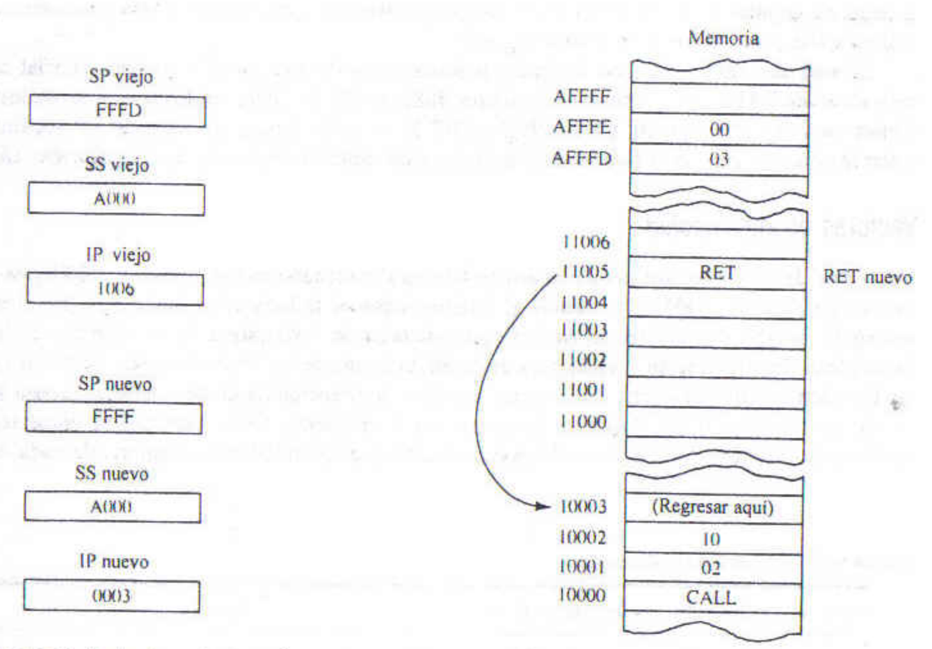


FIGURA 5–8 El efecto de una instrucción RET cercana en la pila y los registros SP, SS e IP.

EJEMPLO 5–11

```
0000                PRUEBA  PROC NEAR

0000 50                     PUSH AX
0001 53                     PUSH BX
                            .    .
                            .    .
0030 C2 0004                RET  4

0033                PRUEBA  ENDP
```

En el ejemplo 5-11 se muestra la forma en que este tipo de retorno borra los datos salvados en la pila por algunas instrucciones PUSH. RET 4 suma 4 a SP antes de recuperar la dirección de retorno de la pila. Debido a que PUSH AX y PUSH BX, juntas, salvaron 4 bytes de datos en la pila, este retorno borra efectivamente a AX y BX de la pila. Este tipo de retorno rara vez aparece en los programas.

5–3 INTRODUCCION A LAS INTERRUPCIONES

Una **interrupción** es una llamada generada por el hardware (derivada en el exterior por una señal de hardware) o una llamada generada por el software (derivada en el interior por una instrucción).

Cualquiera de ellas interrumpirá el programa porque llamará a un *procedimiento para servicio de interrupción o un manejador de interrupción.*

En esta sección se explican las interrupciones por software, que son un tipo especial de las instrucciones CALL en los microprocesadores 8086-80486. Se incluyen los tres tipos de instrucciones para esa interrupción: (INT, INTO e INT 3), se proporciona un mapa de los vectores de interrupción y se explica la finalidad de la instrucción especial de retorno de interrupción (IRET).

Vectores de interrupción

Un **vector de interrupción** es un número de 4 bytes almacenado en los primeros 1024 bytes de la memoria (000000H-0003FFH) cuando el microprocesador trabaja en el modo real. En el modo protegido, la tabla de vectores se sustituye por una tabla de descriptores de interrupción en la que se emplean descriptores de 8 bytes para describir cada una de las interrupciones. Hay 256 vectores de interrupción diferentes. Cada vector contiene la dirección de un procedimiento para servicio de interrupción, o sea el que se llama con una interrupción. En la tabla 5-3 se presentan los vectores de interrupción con una somera descripción y la localidad en la memoria de cada vector

TABLA 5–3 Vectores para interrupción

Número	*Dirección*	*Microprocesador*	*Función*
0	OH–3H	8086–80486	Error al dividir
1	4H–7H	8086–80486	Paso a paso
2	8H–BH	8086–80486	NMI (interrupción de hardware)
3	CH–FH	8086–80486	Punto de ruptura
4	10H–13H	8086–80486	Interrupción por sobreflujo
5	14H–17H	80286–80486	Interrupción de BOUND
6	18H–1BH	80286–80486	Código inválido
7	1CH–1FH	80286–80486	Interrupción de emulación del coprocesador
8	20H–23H	80386–80486	Doble falta
9	24H–27H	80386	Desbordamiento de segmento de coprocesador
10	28H–2BH	80386–80486	Segmento de estado de tarea inválido
11	2CH–2FH	80386–80486	No hay segmento
12	30H–33H	80386–80486	Falla de la pila
13	34H–37H	80386–80486	Falla general de la protección
14	38H–3BH	80386–80486	Falla de página
15	3CH–3FH		Reservado*
16	40H–43H	80286–80486	Error de punto decimal flotante
17	44H–47H	8048–6SX	Interrupción de verificación de alineación
18–31	48H–7FH	8086–80486	Reservado*
32–255	80H–3FFH	8086–80486	Interrupciones por el usuario

* Algunas de estas interrupciones aparecerán en los modelos más nuevos de 8086 a 80486 cuando estén disponibles.

para el modo real. Cada vector contiene un valor para IP y CS que forma la dirección del procedimiento para servicio de la interrupción. Los dos primeros bytes contienen IP y los dos últimos contienen CS.

Intel reserva los primeros 32 vectores de interrupción para los 8086-80486 y futuros productos. Los restantes vectores para interrupción (32 a 255) están disponibles para el usuario. Algunos vectores reservados son para los errores que ocurren durante la ejecución de la programación, tal como la interrupción por error en la división. Algunos vectores están reservados para el coprocesador. Hay otros que ocurren durante los acontecimientos normales en el sistema. En una computadora personal, los vectores reservados se utilizan para las funciones del sistema detalladas más adelante. Los vectores 1 a 6, 7, 9, 16 y 17 funcionan en el modo real y en el modo protegido; los vectores restantes sólo funcionan en el modo protegido.

Instrucciones para interrupción

Los 8086-80486 tienen tres diferentes instrucciones para interrupción disponibles para el programador: INT, INTO e INT 3. En el modo real, cada una de estas instrucciones busca y carga un vector en la tabla de vectores y, luego, llama al procedimiento almacenado en la localidad direccionada por el vector. En el modo protegido, cada una de estas instrucciones busca y carga un descriptor de interrupción de la tabla de descriptores de interrupción. El descriptor especifica la dirección del procedimiento de servicio de interrupción. La llamada a interrupción es similar a una instrucción CALL lejana porque coloca la dirección de retorno (IP, EIP y CS) en la pila.

INT. Hay 256 diferentes instrucciones para interrupción (INT) por software, disponibles para el programador. Cada instrucción INT tiene un operando numérico cuyo intervalo es de 0 a 255 (00H a FFH). Por ejemplo, la INT 100 utiliza el vector 100 de interrupción, que está en las direcciones 190H-193H en la memoria. Para calcular la dirección del vector de interrupción, se multiplica el número del tipo de interrupción por 4. Por ejemplo, la instrucción INT 10H llama al procedimiento de servicio de interrupción cuya dirección está almacenada a partir de la localidad 40H (10H × 4) de la memoria, en el modo real. En el modo protegido, para localizar el descriptor de interrupción se multiplica el número del tipo por 8 en vez de por 4, porque cada descriptor tiene 8 bytes de longitud.

Cada instrucción INT tiene una longitud de 2 bytes. El primer byte contiene el código de operación y el segundo, el número del tipo del vector. La única excepción es INT 3, una interrupción de software especial de un byte, utilizada para los puntos de interrupción.

Siempre que se ejecuta una instrucción de interrupción por software, (1) salva el registro banderas dentro de la pila; (2) desactiva los bits de bandera T e I; (3) salva a CS dentro de la pila; (4) recupera el valor de CS del vector; (5) salva a IP/EIP en la pila; (6) recupera el nuevo valor de IP/EIP del vector; (7) brinca a la nueva localidad direccionada por CS y por IP/EIP. La instrucción INT actúa como una llamada lejana, excepto que no sólo salva a CS e IP en la pila, sino que también salva el registro de banderas en la pila. La instrucción INT es una combinación de PUSHF y de CALL lejana.

Se debe tener en cuenta que cuando se ejecuta la instrucción INT se desactiva la bandera (I) la que controla la terminal de entrada de interrupción externa por hardware INTR (solicitud de interrupción). Cuando I = 0, el microprocesador deshabilita la terminal INTR y, cuando es I = 1, entonces habilita la terminal INTR.

Las interrupciones por software o "blandas" se suelen emplear más para llamar los procedimientos del sistema, los cuales son comunes para todo el software del sistema y de aplicaciones. Las interrupciones, controlan a menudo las impresoras, la exhibición en el monitor y las unidades de disco. La instrucción INT sustituye a una CALL lejana porque la INT tiene longitud de 2 bytes, mientras que la CALL lejana tiene 5 bytes de longitud. Cada vez que la instrucción INT sustituye a una CALL lejana, "ahorra" 3 bytes de memoria en un programa. Esto puede constituir un ahorro considerable si la instrucción INT aparece a menudo en un programa.

IRET/IRETD. La instrucción para retorno de interrupción (IRET) sólo se emplea con procedimientos de servicio de interrupción en el software o en el hardware. Al contrario de una instrucción sencilla (RET) para retorno, la instrucción IRET, hará lo siguiente: (1) recupera datos de la pila y los carga en IP; (2) recupera datos de la pila y los carga en CS; (3) recupera datos de la pila y los carga en el registro de banderas. La instrucción IRET efectúa el mismo trabajo que las instrucciones POPF y RET.

Siempre que se ejecuta una instrucción IRET recupera el contenido original de la banderas I y T de la pila. Esto es importante porque conserva el estado de estos bits de bandera. Si se habilitan las interrupciones antes de un procedimiento de servicio de interrupción, se les vuelve a habilitar automáticamente con la instrucción IRET, porque recupera el contenido original del registro de banderas.

En los 80286 y 80486 se utiliza la instrucción IRET para retornar de un procedimiento de servicio de interrupción al que se llamó en el modo protegido. Es diferente de la IRET porque recupera un apuntador de instrucciones (EIP) de 32 bits de la pila. La IRET se utiliza en el modo real y la IRETD se emplea en el modo protegido.

INT 3. La instrucción INT 3 es una interrupción especial de software destinada a utilizarse como punto de ruptura. La diferencia entre ésta y las otras interrupciones en el software es que INT 3 es una instrucción de 1 byte, mientras que las otras son de 2 bytes.

Se acostumbra introducir una instrucción INT 3 en un programa para interrumpir o romper el flujo. Esta instrucción se llama punto de ruptura y ocurre con cualquier interrupción por software, pero debido a que la INT 3 tiene 1 byte de longitud, es más fácil utilizarla para esta función. Los puntos de ruptura ayudan a depurar programas.

INTO. La instrucción interrupción por sobreflujo (INTO) es una interrupción condicional de software que prueba la bandera de sobreflujo (O). Si O = 0, la instrucción INTO no se ejecuta, pero si O = 1, se ejecuta una interrupción del tipo vector 4.

La instrucción INTO aparece en programas que suman o restan números binarios con signo. Con estas operaciones, es posible tener un sobreflujo. Las instrucciones JO o INTO detectan esta condición de sobreflujo.

Un procedimiento para servicio de interrupción. Supóngase que, en un sistema determinado, hay que sumar el contenido de DI, SI, BP y BX y salvar la suma en AX. Debido a que es una tarea común en ese sistema, vale la pena establecerla como una interrupción por software. En el ejemplo 5-12 se muestra esta interrupción por software. La diferencia principal entre este procedimiento y el procedimiento lejano normal es que termina con la instrucción IRET en vez de la instrucción RET.

EJEMPLO 5-12

```
0000              INTS    PROC FAR

0000 03 C3                ADD  AX,BX
0002 03 C5                ADD  AX,BP
0004 03 C7                ADD  AX,DI
0006 03 C6                ADD  AX,SI
0008 CF                   IRET

0009              INTS    ENDP
```

Control de interrupción

Aunque en este capítulo no se explican las interrupciones por hardware, se presentan dos instrucciones que controlan la terminal INTR del microprocesador. La instrucción para activar la bandera de interrupción (STI) carga un 1 en I, lo cual habilita la terminal INTR. La instrucción para desactivar bandera de interrupción carga un 0 en I, que deshabilita la terminal INTR. La instrucción STI habilita a INTR y la instrucción CLI la deshabilita. En un procedimiento de servicio de interrupción por software, se suelen habilitar las interrupciones por hardware como uno de los primeros pasos, lo cual se logra con la instrucción STI.

Interrupciones en la computadora personal

Las interrupciones que hay en la computadora personal difieren un tanto de las presentadas en la tabla 5-3. La razón de ello es que las computadoras personales originales son sistemas basados en 8086 y 8088. Esto significaba que sólo contenían las interrupciones 0 a 4 especificadas por Intel. Este diseño se ha continuado por lo cual los sistemas más nuevos son compatibles con las primeras computadoras personales.

Debido a que la computadora personal funciona en el modo real, la tabla de vectores de interrupción se encuentra en las direcciones 00000H a 003FFH. Las asignaciones utilizadas por los sistemas de computadoras se presentan en la tabla 5-4. Se debe tener en cuenta que difieren de las asignaciones de la tabla 5-3. Algunas de las instrucciones mostradas en esta tabla se emplean en los programas de ejemplo en capítulos posteriores. Un ejemplo es el tictac (click) de reloj, que es de suma utilidad para temporizar acontecimientos porque ocurre 18.2 veces constantes por segundo en todas las computadoras personales.

Las interrupciones 00H-1FH y 70H-77H siempre están presentes en la computadora, sin que importe el sistema operativo que tengan instalado. Si está instalado el DOS, también estarán presentes las interrupciones 20H-2FH.

5-4 INSTRUCCIONES PARA CONTROL DE MAQUINA Y DIVERSOS

La última categoría de las instrucciones en modo real en los microprocesadores 8086-80486 incluye el grupo de control de maquina y diversos. Estas instrucciones dan el control del bit de acarreo, muestrean la terminal $\overline{\text{BUSY}}$/TEST y efectúan otras funciones. Debido a que muchas de estas instrucciones se emplean en el control del hardware, sólo se explicarán en forma breve en

TABLA 5–4 Asignaciones de interrupciones para la computadora personal

Número	*Función*
0	Error al dividir
1	Paso a paso (depurar)
2	Terminal de interrupción no enmascarable
3	Punto de ruptura
4	Sobreflujo aritmético
5	Imprimir clave de pantalla e instrucción BOUND
6	Error por instrucción ilegal
7	Interrupción por coprocesador no presente
8	Tictac de reloj (hardware) (alrededor de 18.2 Hz)
9	Teclado (hardware)
A	Interrupción 2 de hardware (bus del sistema) (cascada en AT)
B-F	Interrupciones de hardware 3 a 7 en hardware (canal del sistema)
10	BIOS de video
11	Entorno del equipo
12	Memoria de tamaño normal
13	Servicio directo al disco
14	Servicio al puerto COM serial
15	Servicio diverso
16	Servicio al teclado
17	Servicio LPT a puerto paralelo
18	ROM BASIC
19	Borrar y restaurar
1A	Servicio al reloj
1B	Manejador del control de ruptura
1C	Servicio a temporizador del usuario
1D	Apuntador para tabla de parámetros para monitor
1E	Apuntador para tabla de parámetros de unidad de disco
1F	Apuntador para tabla de patrón de caracteres gráficos
20	Terminar el programa
21	Servicios DOS
22	Manejador de terminación del programa
23	Manejador de control C
24	Manejador de error crítico
25	Leer disco
26	Escribir disco
27	Terminar y permanecer residente
28	DOS ocioso
2F	Manejador múltiple
70-77	Interrupciones 8 a 15 en el hardware (computadora estilo AT)

este momento. La mayor parte de estas instrucciones se describen con más detalle en capítulos posteriores relacionados con el hardware y los programas que controlan el hardware.

Control del bit de la bandera de acarreo

La bandera (C) de acarreo propaga el acarreo o el préstamo en sumas y restas de palabras múltiples o de doble palabra. También indica errores en los procedimientos. Hay tres instrucciones que controlan el contenido de la bandera de acarreo: STC (*activar acarreo*), CLC (*desactivar acarreo*) y CMC (*complementar acarreo*).

Debido a que la bandera de acarreo se utiliza muy poco, excepto con sumas y restas de palabras múltiples, está disponible para otros usos. La tarea más común para la bandera de acarreo es indicar un error al retornar de un procedimiento. Suponga que un procedimiento lee datos de un archivo de memoria de disco. Esta operación puede tener éxito o puede ocurrir un error tal como el de no se localizó el archivo. Al retornar de este procedimiento, si C = 1, ha ocurrido un error y si C = 0 no ha ocurrido error. En casi todos los procedimientos de DOS y de BIOS se utiliza la bandera de acarreo para indicar condiciones de error.

WAIT

La instrucción WAIT monitorea la terminal $\overline{\text{BUSY}}$ en el los microprocesadores 80286 y 80386 y la terminal $\overline{\text{TEST}}$ en los 8086 y 8088. El nombre de esta terminal se cambió en el microprocesador 80286, de $\overline{\text{TEST}}$ a $\overline{\text{BUSY}}$. Si se ejecuta la instrucción WAIT mientras la terminal $\overline{\text{BUSY}}$ = 0, no ocurre nada y se ejecuta la siguiente instrucción. Si la terminal $\overline{\text{BUSY}}$ = 1 cuando se ejecuta la instrucción WAIT, el microprocesador espera a que la terminal $\overline{\text{BUSY}}$ regrese a 0 lógico.

La terminal $\overline{\text{BUSY}}$ por lo regular está conectada con la terminal $\overline{\text{BUSY}}$ en los coprocesadores numéricos 8087-80387. Esta conexión, con la instrucción WAIT, permite que los 8086-80386 esperen hasta que el coprocesador termine una tarea. Debido a que el coprocesador está dentro del 80486, no se emplea la terminal $\overline{\text{BUSY}}$.

HLT

La instrucción de detener (HLT) detiene la ejecución del software. Hay tres formas de salir de ella: con una interrupción, con un inicio por hardware o durante una operación de DMA. Esta instrucción suele aparecer en un programa para esperar una interrupción. A menudo sincroniza las interrupciones externas de hardware con el software del sistema.

NOP

Cuando el microprocesador se encuentra con una instrucción (NOP) de no ejecutar operación, sólo necesita un breve tiempo para ejecutarla. Una NOP no efectúa absolutamente ninguna operación y a menudo rellena un programa con un espacio para futuras instrucciones en lenguaje de máquina. Si se están desarrollando programas en lenguaje de máquina, se recomienda colocar la NOP a intervalos de 50 bytes. Esto se hace en caso de que se necesiten agregar instrucciones en el futuro. Una NOP también se puede aplicar en las demoras de tiempo que desperdician periodos cortos de tiempo.

Prefijo LOCK

El prefijo LOCK se agrega al principio de una instrucción y hace que la terminal $\overline{LOCK}$ se convierta en un 0 lógico. $\overline{LOCK}$ a menudo deshabilita a los posibles amos del canal u otros componentes del sistema. El prefijo LOCK hace que la terminal esté activada mientras dura la instrucción con el prefijo LOCK. Si hay más de una instrucción en secuencia, con la terminal $\overline{LOCK}$ sigue siendo un 0 lógico mientras dura la secuencia de las instrucciones. La instrucción LOCK:MOV AL, [SI] es un ejemplo de una instrucción con este prefijo.

ESC

La instrucción para escape (ESC) transmite información al coprocesador numérico 8087 y 80387. Siempre que se ejecuta una instrucción ESC, el microprocesador suministra la dirección en la memoria, si se necesita, pero por lo demás efectúa una NOP. En los 8087-80387 se utilizan 6 bits de la instrucción ESC para obtener el código y empezar a ejecutar una instrucción del coprocesador.

El código de ESC nunca aparece como tal en un programa. En su lugar, hay una serie de instrucciones para el coprocesador (FLD, FST, FMUL, etc.) que ensamblan como instrucciones ESC para el coprocesador. En el capítulo 12 se dan mayores detalles del coprocesador 8087-80287.

BOUND

La instrucción BOUND es para comparar y puede ocasionar una interrupción (vector tipo 5). Esta instrucción compara el contenido de cualquier registro de 16 o de 32 bits contra el contenido de dos palabras o de dobles palabras en la memoria: un límite superior y uno inferior. Si el valor en el registro que se compara con la memoria *no* está dentro de los límites superior e inferior, en seguida aparece una interrupción de tipo 5. Si está dentro del límite, se ejecuta la siguiente instrucción en el programa.

Por ejemplo, si se ejecuta la instrucción BOUND SI,DATO, la localidad DATO, de tamaño palabra, contiene el límite inferior y los bytes DATA + 2 en la localidad de tamaño de palabra, contienen el límite superior. Si el número contenido en SI es menor que el de la localidad DATO en la memoria o mayor que los bytes DATA + 2 en la localidad de memoria, ocurre una interrupción de tipo 5. Se debe tener en cuenta que cuando ocurre la interrupción, la dirección de retorno apunta a la instrucción BOUND, no a la instrucción que sigue a BOUND. Esto difiere de la interrupción normal en la cual la dirección de retorno apunta a la siguiente instrucción en el programa.

ENTER Y LEAVE

Las instrucciones ENTER y LEAVE se utilizan con un cuadro de pila. Un cuadro de pila es un mecanismo utilizado para pasar parámetros a un procedimiento por medio de la memoria de pila. El marco de pila también contiene las variables de memoria local para el procedimiento. Estos marcos producen zonas dinámicas de memoria para los procedimientos en entornos con usuarios múltiples.

La instrucción ENTER crea un marco de pila porque salva a BP dentro de la pila y luego lo carga con la dirección más alta del marco de pila. Esto permite accesar a las variables en el marco

de pila mediante el registro BP. La instrucción ENTER contiene dos operandos: el primero especifica el número de bytes que se deben reservar para las variables en el marco de pila y el segundo especifica el grado o categoría del procedimiento.

Suponga que se ejecuta una instrucción ENTER 8.0, la cual reserva 8 bytes de memoria para el marco de pila y el cero especifica grado 0. En la figura 5-9 se muestra el marco de pila establecido con esta instrucción. Se debe tener en cuenta que la instrucción almacena a BP en la parte superior de la pila. Después resta 8 en el apuntador de pila y deja 8 bytes de espacio en la memoria para el almacenamiento temporal de datos. La localidad más alta de esta zona de 8 bytes para almacenamiento temporal, está direccionada por BP. Con la instrucción LEAVE se invierte este proceso porque se vuelven a cargar SP y BP con los valores que tenían antes.

EJEMPLO 5–13

```
                         ;secuencia utilizada para llamar al programa del sistema que
                         ;emplea los parámetros almacenados en un marco de pila
                         ;
0000 C8 0004 00          ENTER 4,0                  ;crear un marco de 4 bytes
0004 A1 00C8 R           MOV   AX,DATA1
0007 89 46FC             MOV   [BP-4],AX            ;salvar parámetro 1
000A A1 00CA R           MOV   AX,DATA2
000D 89 46FE             MOV   [BP-2],AX            ;salvar parámetro 2

0010 E8 0100 R           CALL  SYS                  ;llamar la subrutina

0013 8B 46FC             MOV   AX,[BP-4]            ;obtener resultado 1
0016 A3 00C8 R           MOV   DATA1,AX             ;salvar resultado 1
0019 8B 46FE             MOV   AX,[BP-2]            ;obtener resultado 2
001C A3 00CA R           MOV   DATA2,AX             ;salvar resultado 2
```

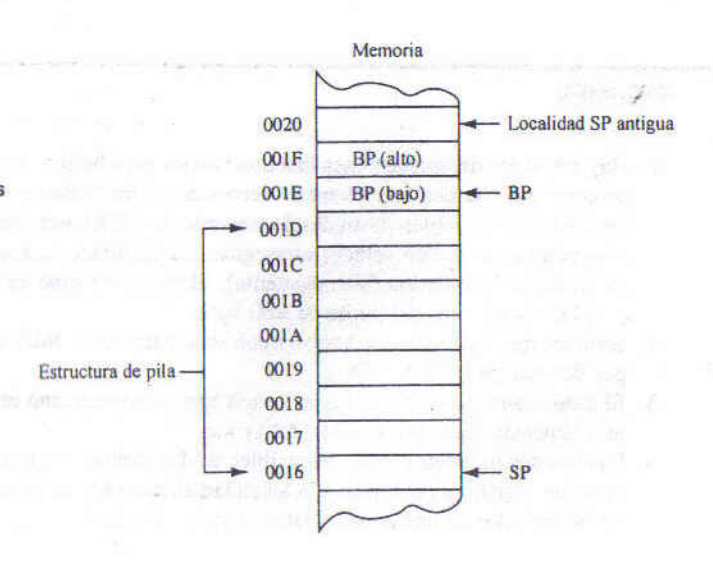


FIGURA 5–9 Una estructura de pila generada por la instrucción ENTER 8,0. El registro BP se guarda empezando por la parte superior de la estructura de pila. A esto le sigue un área de 8 bytes que se denomina estructura de pila.

```
001F C9                  LEAVE
                          .
                          .
                         (otro programa continúa aquí)
                          .
                          .
                ;subrutina del sistema que utiliza los parámetros del marco de pila
                ;
0100            SYS      PROC   NEAR

0100 60                  PUSHA

0101 8B 46 FC            MOV    AX,[BP-4]            ;obtener parámetro 1
0104 8B 5E FE            MOV    BX,[BP-2]            ;obtener parámetro 2
                          .      .
                          .      .
                         (programa que utiliza los parámetros)
                          .

0130 89 46 FC            MOV    [BP-4],AX            ;salvar resultado 1
0133 89 5E FE            MOV    [BP-2],BX            ;salvar resultado 2

0136 61                  POPA
0137 C3                  RET

0138            SYS      ENDP
```

En el ejemplo 5-13 se muestra la forma en que la instrucción ENTER produce un marco de pila a fin de poder pasar dos parámetros de 16 bits a un procedimiento del sistema con un nivel. Se debe tener en cuenta cómo aparecen las instrucciones ENTER y LEAVE en este programa y la forma en que los parámetros pasan a través del marco de pila hacia y desde el procedimiento. En este procedimiento se emplean 2 parámetros que lo pasan al marco de pila y retornan 2 resultados desde el marco.

5–5 RESUMEN

1. Hay tres tipos de instrucciones incondicionales para brinco: corto, cercano y lejano. El brinco corto permite una transferencia dentro de un intervalo de +127 y −128 bytes. El brinco cercano (con el empleo de un desplazamiento de ±32K) permite un brinco a cualquier sitio en el segmento de código actual (intrasegmento). El brinco lejano permite un salto a cualquier localidad en la memoria (intersegmento). El brinco cercano en los microprocesadores 80386 y 80486 está dentro del límite de ±2G bytes.
2. Siempre que aparezca una etiqueta con una instrucción JMP, la etiqueta debe estar seguida por dos puntos (ETIQUETA:).
3. El desplazamiento que sigue a un brinco corto o uno cercano es la distancia desde la siguiente instrucción hasta la localidad del brinco.
4. Los brincos indirectos están disponibles en dos formas: (1) brinco a una localidad almacenada en un registro y (2) brinco a la localidad almacenada en la memoria en una palabra (cercano indirecto) o en una doble palabra (lejano indirecto).

5. Los brincos condicionales son todos cortos y prueban uno o más de los bits de bandera C, Z, O, P o S. Si la condición es verdadera, ocurre el brinco y si la condición es falsa, se ejecuta la siguiente instrucción en el orden.
6. Una instrucción especial para brinco condicional (LOOP) decrementa CX y brinca a la etiqueta cuando CX no es 0. Otras formas de LOOP incluyen LOOPE, LOOPNE, LOOPZ y LOOPNZ. La instrucción LOOPE produce el brinco si CX no es cero y si existe una condición de igualdad. En los microprocesadores 80386 y 80486 la instrucción LOOP también puede utilizar al registro ECX como contador.
7. En los microprocesadores 80386 y 80486 hay un grupo de instrucciones de activación, de acuerdo con las condiciones que hacen un byte 01H o 00H. Si la condición sometida a prueba es verdadera, se establece el byte del operando a 01H y si la condición que se prueba es falsa, el byte del operando se borra a 00H.
8. Los procedimientos son grupos de instrucciones que efectúan una tarea y se utilizan desde cualquier punto en un programa. La instrucción CALL se liga con un procedimiento y la instrucción RET retorna desde aquél. En el lenguaje ensamblador, el directivo PROC define el nombre y tipo del procedimiento. El directivo ENDP declara el final del procedimiento.
9. La instrucción CALL es una combinación de una instrucción PUSH y una JMP. Cuando se ejecuta CALL, salva la dirección de retorno dentro de la pila y, luego, brinca al procedimiento. Una CALL cercana salva el contenido de IP en la pila y una CALL lejana salva a IP y a CS en la pila.
10. La instrucción RET, para retornar de un procedimiento, recupera la dirección de retorno de la pila y la carga en IP (retorno cercano) o en IP y CS (retorno lejano).
11. Las interrupciones "blandas" son instrucciones similares a CALL o señales del hardware utilizadas para llamar a procedimientos. Este proceso interrumpe el programa en curso y llama a un procedimiento. Después del procedimiento, una instrucción IRET retorna el control al programa interrumpido.
12. Los vectores de interrupción tienen 4 bytes de longitud que contienen la dirección (IP y CS) del procedimiento de servicio de la interrupción. El microprocesador contiene 256 vectores de interrupción en el primer 1K byte de memoria. Los primeros 32 los define Intel; los 224 restantes son para interrupción por el usuario.
13. Siempre que el microprocesador acepta una interrupción, se salvan las banderas, IP y CS dentro de la pila. Además de salvar a las banderas se desactivan los bits de bandera I y T para deshabilitar la función de trazo y a la terminal INTR. La activa final que ocurre para la interrupción es que se usa y se carga el vector de interrupción de la tabla de vectores y ocurre un brinco al procedimiento de servicio de la interrupción.
14. Las instrucciones (INT) para interrupción por software, sustituyen a menudo a las llamadas al sistema. Las interrupciones de software ahorran 3 bytes de memoria cada vez que sustituyen a las instrucciones CALL.
15. Se debe utilizar una instrucción (IRET) especial para retorno para regresar de un procedimiento de servicio de interrupción. La instrucción IRET no sólo recupera a IP y CS de la pila, también las banderas.
16. Interrumpir por un sobreflujo (INTO) es una interrupción condicional que llama a un procedimiento de servicio de interrupción, si la bandera (O) de sobreflujo = 1.
17. La bandera (I) de habilitación de interrupción controla la conexión en la terminal INTR del microprocesador. Si se ejecuta la instrucción STI, activa a I para habilitar la terminal INTR. Si se ejecuta la instrucción CLI, desactiva (0) a I para deshabilitar la terminal INTR.

18. El bit (C) de la bandera de acarreo se desactiva, activa y complementa con las instrucciones CLC, STC y CMC.
19. La instrucción WAIT prueba la condición de la terminal $\overline{BUSY}$ en el microprocesador. Si $\overline{BUSY}$ = 0, WAIT no espera, pero si $\overline{BUSY}$ = 1, WAIT continúa con la prueba de la terminal $\overline{BUSY}$ hasta que sea un 0 lógico.
20. El prefijo LOCK hace que la terminal $\overline{LOCK}$ se vuelva un 0 lógico mientras dure la instrucción siguiente. La instrucción ESC pasa instrucciones al coprocesador numérico 80287 y 80387.
21. La instrucción BOUND compara el contenido de cualquier registro de 16 bits contra el contenido de dos palabras en la memoria: un límite superior y uno inferior. Si el valor en el registro comparado con la memoria *no* está dentro de los límites superior e inferior, ocurre una interrupción de tipo 5.
22. Las instrucciones ENTER y LEAVE se utilizan con los marcos de pila. Un marco de pila es un mecanismo utilizado para pasar parámetros a un procedimiento por medio de la memoria de pila. El marco de pila también contiene variables locales de la memoria para el procedimiento. La instrucción ENTER produce el marco de pila; la instrucción LEAVE extrae el marco de la pila. El registro BP direcciona los datos en el marco de pila.

5–6 CUESTIONARIO Y PROBLEMAS

1. ¿Qué es un brinco o JMP corto?
2. ¿Qué tipo de JMP se utiliza cuando se brinca a cualquier lugar dentro de un segmento?
3. ¿Cuál instrucción JMP permite al programa continuar la ejecución en cualquier localidad de memoria en el sistema?
4. ¿Cuál instrucción JMP tiene 5 bytes de longitud?
5. ¿Cuál es el alcance de un brinco cercano en los microprocesadores 80386 y 80486?
6. ¿Qué se puede decir acerca de una etiqueta que va seguida por dos puntos (:)?
7. ¿Qué registro o registros cambia el brinco cercano para modificar la dirección del programa?
8. ¿Qué registro o registros cambia el brinco lejano para modificar la dirección del programa?
9. Explique lo que logra la instrucción JMP AX. Aclare también si es instrucción para salto cercano o lejano.
10. Compare el funcionamiento de una JMP DI con una JMP [DI].
11. Compare el funcionamiento de una IMP [DI] con una JMP FAR PTR [DI].
12. Enumere los cinco bits de bandera que se prueban con las instrucciones condicionales para brinco.
13. Describa cómo funciona la instrucción JA.
14. ¿Cuándo brincará la instrucción JO?
15. ¿Qué instrucciones para salto condicional siguen a la comparación de los números con signo?
16. ¿Qué instrucciones para salto condicional siguen a la comparación de números sin signo?
17. ¿Qué instrucciones para salto condicional prueban los bits Z y C de bandera?
18. ¿Cuándo brinca la instrucción JCXZ?
19. ¿Qué instrucción SET se utiliza para activar a AL si los bits de bandera indican una condición de cero?

20. La instrucción LOOP en el 8086 decrementa el registro de __________ y lo prueba para ver si hay un 0 para decidir si ocurre un brinco.
21. La instrucción LOOP en el 80486 decrementa el registro __________ y lo prueba para ver si hay un 0 para decidir si ocurre un brinco.
22. Explique cómo funciona la instrucción LOOPE.
23. Escriba una serie corta de instrucciones que almacene un 00h en 150H bytes de memoria, a partir de la localidad DATO en la memoria del segmento extra. Hay que emplear la instrucción LOOP para llevar a cabo esta tarea.
24. Escriba una secuencia de instrucciones que rastree en un bloque de 100H bytes de memoria. Este programa debe contar todos los números sin signo que están por arriba y por abajo de 42H. La localidad ALTO de tamaño de byte en la memoria debe contener el conteo de los números por arriba de 42H y la localidad BAJO debe contener el conteo de los números que hay por debajo de 42H.
25. ¿Qué es un procedimiento?
26. Explique cómo funcionan las instrucciones CALL cercana y lejana.
27. ¿Cómo funciona la instrucción RET cercana?
28. La última instrucción ejecutable en un procedimiento debe ser una__________.
29. ¿Con qué directivo se designa el comienzo de un procedimiento?
30. ¿Cómo se determina si un procedimiento es cercano o lejano?
31. Explique lo que logra la instrucción RET 6.
32. Escriba un procedimiento cercano que eleve al cubo el contenido del registro CX. Este procedimiento no puede cambiar ningún registro excepto CX.
33. Escriba un procedimiento que multiplique DI por SI y, luego, divida el resultado entre 100H. Compruebe que el resultado se queda en AX cuando ocurra el retorno del procedimiento. Este procedimiento no puede cambiar ningún registro excepto AX.
34. Redacte un procedimiento que sume EAX, EBX, ECX y EDX. Si ocurre un acarreo, ponga un 1 lógico en EDI. Si no ocurre acarreo ponga un 0 en EDI. La suma debe quedar en EAX después de la ejecución del procedimiento.
35. ¿Qué es una interrupción?
36. ¿Cuáles instrucciones de software llaman a un procedimiento de servicio de interrupción?
37. ¿Cuántos tipos diferentes de interrupciones están disponibles en el microprocesador?
38. ¿Cuál es la finalidad del vector tipo 0 para interrupción?
39. Ilustre el contenido de un vector de interrupción y explique la finalidad de cada parte.
40. ¿En qué difiere la instrucción IRET de la instrucción RET?
41. ¿Qué es la instrucción IRETD?
42. ¿Sólo en qué condiciones interrumpe la instrucción INTO el programa?
43. ¿En qué localidades de memoria está almacenado el vector de interrupción para una instrucción INT 40H?
44. ¿Qué instrucciones controlan el funcionamiento de la terminal INTR?
45. ¿Qué interrupción en una computadora personal da servicio al puerto paralelo LPT?
46. ¿Qué interrupción en una computadora personal da servicio al teclado?
47. ¿Qué instrucción prueba la terminal BUSY?
48. ¿Cuándo interrumpirá un programa la instrucción BOUND?
49. Una instrucción ENTER 16.0 produce un marco de pila que contiene __________ bytes.
50. ¿Qué registro mueve a la pila cuando se ejecuta una instrucción ENTER?
51. ¿Qué instrucción pasa los opcodes al coprocesador numérico?

CAPITULO 6

Programación del microprocesador

INTRODUCCION

En este capítulo se presentan programas y técnicas para programación con el empleo del programa macro ensamblador MASM, las llamadas a funciones del DOS, y las llamadas a funciones del BIOS. En el capítulo se utilizan algunas de las llamadas a las funciones del DOS y a las funciones del BIOS, pero todas se explican con todo detalle en el apéndice A. Repase las llamadas a funciones según lo requiera, conforme lea el capítulo. El ensamblador MASM ya ha quedado explicado y utilizado en capítulos anteriores, pero aquí se presentan todavía más características.

Algunas de las técnicas de programación que se explican en este capítulo incluyen: secuencias de macros, manejo del teclado y el monitor, módulos de programa, archivos de biblioteca, interrupciones, y otras importantes técnicas para programación. Este capítulo es una introducción a la programación, pero incluye técnicas valiosas para programación que serán de enorme ayuda para poder desarrollar programas para computadora personal con el empleo de PCDOS[1] y MSDOS[2] como punto de partida.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Usar el ensamblador MASM y el programa ligador para producir programas que contengan más de un módulo.
2. Explicar el empleo de EXTRN y PUBLIC y cómo se aplican en la programación modular.
3. Inicializar un archivo de biblioteca que contenga subrutinas de uso común.
4. Escribir y utilizar MACRO y ENDM para crear las macrosecuencias utilizadas —con la programación lineal.

[1] PCDOS es marca registrada de IBM Corporation.

[2] MSDOS es marca registrada de Microsoft Corporation.

5. Mostrar la forma en que se desarrollan archivos para acceso secuencial y aleatorio para empleo en un sistema.
6. Desarrollar programas con llamadas a funciones del DOS.
7. Desarrollar la diferencia entre una llamada a funciones del DOS y a una función del BIOS.
8. Mostrar la forma de cambiar el servicio de las interrupciones con el empleo de llamadas a funciones del DOS.

6–1 PROGRAMACION MODULAR

Muchos programas son demasiado largos para que los redacte una sola persona. Esto significa que a veces los programas los redactan equipos de programadores. El programa ligador con que cuentan MSDOS y PCDOS permite que los módulos de programación se puedan ligar entre sí para formar un programa completo. En este capítulo se describen el ligador, la tarea de ligamiento, los archivos de biblioteca, EXTRN y PUBLIC según se aplican a los módulos y la programación modular.

El ensamblador y el ligador

El *programa ensamblador* convierte un módulo simbólico fuente (archivo) en un archivo objeto hexadecimal. En capítulos anteriores se han visto muchos ejemplo de archivos simbólicos fuente. En el ejemplo 6-1 se muestra el diálogo con el ensamblador que aparece cuando se ensambla un módulo fuente llamado ARCH.ASM. Siempre que se crea un archivo fuente debe tener una extensión ASM. Se debe tener en cuenta que la extensión (.ASM) no se teclea en el indicador del ensamblador al ensamblar un archivo. Para crear los archivos de fuente se emplea un editor que viene con el ensamblador o con casi cualquier otro procesador de palabras o editor que pueda generar un archivo ASCII.

EJEMPLO 6–1

```
A>MASM

Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Microsoft Corp 1981, 1989. All rights reserved.

Nombre de archivo fuente [ .ASM] : ARCHIVO
Nombre de archivo objeto [ FILE. OBJ] : ARCHIVO
Listado fuente [ NULLST] : ARCHIVO
Referencia cruzada [ NUL.CRF] : ARCHIVO
```

El programa ensamblador (MASM) solicita el nombre del archivo fuente, el nombre del archivo objeto, el nombre para el listado del archivo y un nombre de archivo para referencia cruzada. El archivo objeto (OBJ) no es ejecutable pero está designado como el archivo de entrada al ligador. El archivo para el listado fuente (LST) contiene la versión ensamblada del archivo fuente y su equivalente en lenguaje de máquina hexadecimal. El archivo de referencia cruzada (CRF) incluye todas las etiquetas y la información pertinente requerida para la referencia cruzada.

El programa ligador lee los archivos objeto, creados por el programa ensamblador y los liga para formar un solo archivo para ejecución. El *archivo para ejecución* se crea con la extensión EXE del nombre del archivo. Para ejecutar estos archivos se teclea el nombre del archivo en el

indicador de DOS (A>). Un ejemplo de archivo para ejecución es el RANA.EXE y para ejecutarlo se teclea RANA en el indicador de comando de DOS.

Si el archivo es lo suficiente corto, de menos de 64K bytes de longitud, se puede convertir de archivo para ejecución a *archivo para comando* (COM). El archivo para comando es un poco diferente de un archivo para ejecución en que el programa se debe originar a partir de la localidad 100H antes de que se pueda ejecutar. El programa EXE2BIN se emplea para convertir un archivo para ejecución en archivo para comando. La ventaja principal de un archivo para comando es que descarga el disco hacia la computadora con mucha más rapidez que un archivo para ejecución. También requiere mucho menos espacio para almacenamiento en disco que el archivo equivalente para ejecución.

EJEMPLO 6–2

```
A>LINK

Microsoft (R) Overlay Linker Version 3.64
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.

Módulos objeto [ .OBJ] : RANA+QUE+DONA
Archivo ejecutable [ RANA.EXE] : RANA
Archivo listado [ NUL.MAP] : RANA
Bibliotecas [ .LIB] : LIBS
```

En el ejemplo 6-2 se muestra el protocolo incluido en el programa ligador cuando se utiliza para ligar los archivos RANA, QUE y DONA. El ligador también liga los archivos de biblioteca (LIBS) a fin de poder utilizar los procedimientos, ubicados dentro de LIBS con el programa para ejecución ligado. Para llamar al ligador se teclea LINK en el indicador de comando de DOS, como se muestra en el ejemplo 6-2. Se debe tener en cuenta que antes de ligar los archivos primero se deben ensamblar y deben estar *libres de error.*

En este ejemplo, después de teclear LINK, el programa ligador pregunta por los "Módulos objeto" que produce el ensamblador. En este ejemplo hay tres módulos objeto: RANA , QUE y DONA. Si hay más de un archivo objeto, se teclea primero el programa principal (RANA en este ejemplo) seguido por cualesquiera otros módulos de apoyo. (Se utiliza un signo positivo (+) para separar los nombres de los módulos.)

Después de teclear los nombres de los módulos del programa, el ligador sugiere que el nombre del archivo ejecutable sea RANA.EXE. Para lograrlo, se teclea el mismo nombre o si se desea, se oprime la tecla Enter. También se puede cambiar a cualquier otro nombre en este momento, si se desea.

El archivo para el listado es un mapa en donde aparecen los segmentos del programa producidos por el ligamiento. Si se oprime la tecla Enter no se produce archivo para el listado, pero si se teclea el nombre, el archivo para el listado aparece en el disco.

A los archivos de biblioteca se les da entrada en la última línea. En este ejemplo se empleó el nombre LIBS del archivo de biblioteca. Esta biblioteca contiene procedimientos utilizados en los otros módulos de programa.

PUBLIC y EXTRN

Los directivos PUBLIC y EXTRN son muy importantes en la programación modular. Se utilizan para declarar que las etiquetas en códigos, el segmento de datos o en todos los segmentos están

disponibles para otros módulos de programa. Se utiliza EXTRN (externas) para declarar que las etiquetas son externas a un módulo. Sin estos enunciados, no se pueden ligar los módulos entre sí para producir un programa con el empleo de técnicas de programación modular.

EJEMPLO 6–3

```
                    DAT1   SEGMENT PUBLIC           ;declarar público todo el segmento

                           PUBLIC DATO1             ;declarar públicos a DATO1, DATO2
                           PUBLIC DATO2

0000 0064[          DATO1  DB     100 DUP (?)       ;global
           ??
              ]
0064 0064[          DATO2  DB     100 DUP (?)       ;global
           ??
              ]
00C8                DAT1   ENDS

0000                CODES  SEGMENT

                           ASSUME CS:CODES,DS:DAT1

                           PUBLIC LEE               ;declarar público a LEE

0000                LEE    PROC FAR

0000 B4 06                 MOV  AH,6                ;leer teclado
0002 B2 FF                 MOV  DL,0FFH             ;sin eco
0004 CD 21                 INT  21H
0006 74 F8                 JE   LEE
0008 CB                    RET

0009                LEE    ENDP

0009                CODES  ENDS

                           END
```

El directivo PUBLIC se suele colocar en el campo del código de un enunciado en lenguaje ensamblador para definir que una etiqueta es pública, a fin de poder emplearla en otros módulos. Esta etiqueta puede ser una dirección para brinco, una dirección de datos o todo un segmento que se pueden hacer públicos. En el ejemplo 6-3 se muestra el enunciado PUBLIC utilizado para definir que algunas etiquetas son públicas para otros módulos. Cuando los segmentos se hacen públicos se combinan con otros segmentos públicos que contienen datos con el mismo nombre del segmento.

La declaración EXTRN aparece en los segmentos de datos y de código para definir que las etiquetas son externas al segmento. Si se define que los datos son externos su tamaño se debe expresar como, BYTE, WORD (Palabra) o DWORD. Si una dirección para brinco o para llamada son externas, se deben representar como Cercana (NEAR) o Lejana (FAR). En el ejemplo 6-4 se muestra la forma en que se emplea la declaración de Externo para indicar que cierto número de etiquetas son externas al programa que se enlista. Se debe tener en cuenta que en este ejemplo, cualquier dirección o datos externos se definen con la letra E en la lista hexadecimal del ensamblador.

EJEMPLO 6–4

```
0000                DATO1  SEGMENT PUBLIC          ;declarar público todo el segmento

                           EXTRN DATO1:BYTE
                           EXTRN DATO2:BYTE
                           EXTRN DATO3:WORD
                           EXTRN DATO4:DWORD

0000                DATO1  ENDS

0000                CODES  SEGMENT

                           ASSUME CS:CODES,ES,DAT1

                           EXTRN LEE:FAR

0000                MAIN   PROC  FAR

0000 B8--R                 MOV   AX,DATO1
0003 8E C0                 MOV   ES,AX

0005 BF 0000 E             MOV   DI,OFFSET DATO1
0008 B9 000A               MOV   CX,10

000B                PRINC1:

000B 9A 0000--E            CALL  LEE
0010 AA                    STOSB
0011 E2 F8                 LOOP  PRINC1
0013 CB                    RET

0014                MAIN   ENDP

0014                CODES  ENDS

                           END   MAIN
```

Bibliotecas

Los archivos de bibliotecas son colecciones de procedimientos que se pueden emplear en muchos programas diferentes. Estos procedimiento se ensamblan y se compilan en un archivo de biblioteca con el programa LIB que acompaña al programa ensamblador MASM. Las bibliotecas permiten acopiar procedimientos comunes en un lugar, a fin de que se puedan emplear en muchas aplicaciones diferentes. El archivo de biblioteca se indica cuando se liga un programa con el ligador.

¿Por qué batallar con archivos de biblioteca?, un archivo de biblioteca es un buen lugar para almacenar una colección de procedimientos relacionados. Cuando un archivo de biblioteca se liga con un programa, solamente los procedimientos requeridos por el programa son tomados de los archivos de la biblioteca y agregados al programa. Si se ha de lograr con eficiencia cualquier cantidad de programación en lenguaje ensamblador es esencial un buen conjunto de archivos de biblioteca.

Cómo crear un archivo de biblioteca. Un archivo de biblioteca se crea con el comando LIB tecleado en el indicador de DOS. Un archivo de bilbioteca es una colección de archivos. OBJ ensamblados en que cada uno desempeña un procedimiento o tarea. En el ejemplo 6-5 se ilustra la forma en

que se utilizarán dos archivos separados (LEE_TECLA y ECO) para estructurar un archivo de biblioteca. Se debe tener en cuenta que el nombre del procedimiento se debe declarar en un archivo de biblioteca y no necesita tener el nombre del archivo, aunque sí lo tiene el siguiente ejemplo.

EJEMPLO 6–5

```
                              ;El primer módulo de biblioteca se llama LEE_TEC.
                              ;Este procedimiento lee una tecla del teclado
                              ;y retorna con los caracteres en ASCII en AL
                              ;
0000                          LIB      SEGMENT

                                       ASSUME CS:LIB

                                       PUBLIC LEE_TEC

0000                          LEE_TEC        PROC    FAR

0000 52                                      PUSH    DX

0001                          LEE_TEC1:
0001 B4 06                                   MOV     AH,6
0003 B2 0F                                   MOV     DL,0FH
0005 CD 21                                   INT     21H
0007 74 F8                                   JE      LEE_TEC1:
0009 5A                                      POP     DX
000A CB                                      RET
000B                          LEE_TEC        ENDP

000B                          LIB      ENDS

                                       END
                              ;
                              ;Este segundo módulo de biblioteca se llama ECO
                              ;este procedimiento exhibe los caracteres en ASCII
                              ;en AL en la pantalla.
                              ;
0000                          LIB      SEGMENT
                                       ASSUME CS:LIB

                                       PUBLIC ECO

0000                          ECO            PROC    FAR
0000 52                                      PUSH    DX
0001 B4 06                                   MOV     AH,6
0003 8A D0                                   MOV     DL,AL
0005 CD 21                                   INT     21H
0007 5A                                      POP     DX
0008 CB                                      RET
0009                          ECO            ENDP

0009                          LIB      ENDS

                                       END
```

Después de ensamblar cada archivo, se utiliza el programa LIB para combinarlos en un archivo de biblioteca. El programa LIB indica que se necesita información como se ilustra en el ejemplo 6-6, en el cual estos archivos se combinaron para formar la biblioteca IO.

EJEMPLO 6–6

```
A>LIB

Microsoft (R) Library Manager Version 3.10
Copyright (C) Microsoft Corp 1983-1988. All rights reserved.
Nombre de la biblioteca:IO
No hay archivo de biblioteca. ¿Abrirlo? Sí
Operaciones: LEE_TEC+ECO
Archivo de listado:IO
```

El programa LIB empieza con el aviso de propiedad literaria (copyright) de Microsoft, seguido por la indicación *Library name* (nombre de la biblioteca). El nombre de la biblioteca seleccionado es IO para el archivo IO.LIB. Debido a que se trata de un archivo nuevo, el programa de biblioteca pregunta si se desea crear el archivo de biblioteca. En indicador *Operations* se teclea el nombre del nuevo módulo de biblioteca. En este caso se creó una biblioteca con el empleo de dos archivos de procedimiento (LEE_TECLA y ECO). El archivo del listado muestra el contenido de la biblioteca y se ilustra para esta biblioteca en el ejemplo 6-7. El archivo del listado muestra el tamaño y nombres de los archivos utilizados para producir la biblioteca y también la etiqueta de Público (nombre del procedimiento) que se empleó en el archivo de biblioteca.

EJEMPLO 6–7

```
ECO .......ECO              LEE_TEC ......LEE_TEC

LEE_TEC       Offset: 00000010H Tamaño de código y datos: BH
 LEE_TEC

ECO           Offset: 00000070H Tamaño de código y datos: 9H
 ECO
```

Si después hay que agregar módulos adicionales de biblioteca, teclee el nombre del archivo de biblioteca después de haber utilizado LIB. En *Operations*, teclee el nuevo nombre del módulo precedido por el *signo* + para agregar un nuevo procedimiento. Si se tiene que suprimir un módulo de biblioteca utilice el *signo negativo (-)* antes de la operación del nombre del archivo.

Una vez que el archivo de biblioteca está ligado con el archivo del programa, sólo los procedimientos de la biblioteca que en realidad se vayan a emplear se colocan en el archivo de ejecución. No olvide utilizar la etiqueta EXTRN al especificar llamadas a la biblioteca desde el módulo de su programa.

Macros

Un macro es un grupo de instrucciones que efectúan una tarea, igual que el procedimiento ejecuta una sola tarea. La diferencia es que a un procedimieto se accede con una instrucción CALL, mientras que el macro se inserta en el programa como nuevo código que contiene una secuencia de instrucciones. Un macro es un nuevo código que usted produce. Las macrosecuencias se ejecutan con mayor rapidez que los procedimientos porque no hay que ejecutar instrucciones CALL o RET. Las instrucciones del macro se colocan en el programa en el punto en que se les "invoca".

Los directivos MACRO y ENDM se emplean para delinear una macrosecuencia. El primer enunciado de un macro es el enunciado MACRO que contiene el nombre de la macro y cuales-

quiera parámetros relacionados con ello. Un ejemplo es TRANSF MACRO A,B que define que el macro es TRANSF. En este nuevo *código* se emplean dos parámetros: A y B. El último enunciado de un macro es la instrucción ENDM que es una línea por sí sola.

EJEMPLO 6–8

```
                     TRANSF MACRO  A,B          ;;mueve palabra desde B hasta A

                            MOV    AX,B
                            MOV    A,AX
                            POP    AX

                            ENDM

                            MOVE   VAR1,VAR2    ;usar macro TRANSF

0000 50            1        PUSH   AX
0001 A1 0002 R     1        MOV    AX,VAR2
0004 A3 0000 R     1        MOV    VAR1,AX
0007 58            1        POP    AX

                            MOVE   VAR3,VAR4    ;usar macro TRANSF

0008 50            1        PUSH   AX
0009 A1 0006 R     1        MOV    AX,VAR4
000C A3 0004 R     1        MOV    VAR3,AX
000F 58            1        POP    AX
```

En el ejemplo 6-8 se muestra cómo se crea un macro y cómo se utiliza en un programa. Este macro transfiere el contenido de tamaño de palabra de la localidad B de la memoria y a la localidad A de la memoria de tamaño de palabra. Después de que se ha definido el macro, este ejemplo se utiliza dos veces. El macro está *expandido* en este ejemplo, a fin de que se pueda ver la forma en que se ensambla para generar las transferencias. Cada instrucción hexadecimal en lenguaje de máquina seguida por un número (un 1 en este ejemplo), es una expansión de la declaración del macro. La declaración de ampliación no se tecleó en el programa fuente. Se debe tener en cuenta que el comentario en la macro va precedido por a;; en vez de ; como se acostumbra.

Variable local en un macro. A veces, los macros contienen variables locales. Una *variable local* es la que aparece en el macro, pero no está disponible fuera de él. Para definir una variable local se utiliza el directivo LOCAL. En el ejemplo 6-9 se muestra la forma en que una variable local, utilizada como dirección para brinco aparece en una definición de macro. Si esta dirección para brinco no está definida como local, el ensamblador mostrará una bandera de error en el segundo y en los subsecuentes intentos por usar el macro.

EJEMPLO 6–9

```
LEE    MACRO A             ;;lee teclado
       LOCAL READ1         ;;define a LEE1 como local

       PUSH  DX

LEE1:
```

```
                                MOV     AH,6
                                MOV     DL,0FFH
                                INT     21H
                                JE      LEE1
                                MOV     A,AL
                                POP     DX

                                ENDM

                                LEE     VAR5                    ;leer tecla

0000 52             1           MOV     DX

0001                1   ??0000:
0001 B4 06          1           MOV     AH,6
0003 B2 FF          1           MOV     DL,0FFH
0005 CD 21          1           INT     21H
0007 74 F8          1           JE      ??0000
0009 A2 0008 R      1           MOV     VAR5,AL
000C 5A             1           POP     DX

                                LEE     VAR6                    ;leer tecla

000D 52             1           PUSH    DX
000E                1   ??0001:
000E B4 06          1           MOV     AH,6
0010 B2 FF          1           MOV     DL,0FFH
0012 CD 21          1           INT     21H
0014 74 F8          1           JE      ??0001
0016 A2 0009 R      1           MOV     VAR6,AL
0019 5A             1           POP     DX
```

En este ejemplo, se lee un carácter del teclado y se almacena en la localidad de memoria de tamaño de byte indicada como parámetro en el macro. Se debe tener en cuenta la forma en que la etiqueta LEE1 en los macros expandidos.

El directivo LOCAL siempre debe seguir de inmediato al directivo MACRO sin que haya ningún espacio ni texto entre ellas. Si aparecen un espacio o un texto entre un MACRO y LOCAL, el ensamblador indica un error y no aceptará a la variable como local.

Colocación de MACRO definiciones en su propio módulo. Las macrodefiniciones se pueden colocar en el archivo del programa como se ilustra, o se pueden colocar en su propio macromódulo. Es posible producir un archivo que contenga sólo los macros que se van a incluir en otros archivos de programa. Se utiliza el directivo INCLUDE para indicar que un archivo de programa incluirá un módulo que contenga macrodefiniciones externas. Aunque no es un archivo de biblioteca, en la práctica funciona como biblioteca de macrosecuencias.

Cuando las secuencias macro se colocan en un archivo (a menudo con extensión INC o MAC) no contienen enunciados PUBLIC. Si un archivo llamado MACRO.MAC contiene macro secuencias, el enunciado INCLUDE se coloca en el archivo del programa como INCLUDE C:\ASSM\MACRO.MAC. Se debe tener en cuenta que el archivo del macro está en la unidad de disco C, el subdirectorio ASSM en este ejemplo. La declaración INCLUDE incluye estos macros igual que si se hubieran tecleado en el archivo. No se necesita un enunciado EXTRN para acceder a las macro declaraciones que se han incluido.

La aproximación de la programación modular

La aproximación de la programación modular, incluye a menudo un equipo de personas a quienes se asignan diferentes tareas de programación. Esto permite que el director del equipo pueda asignar partes del programa a diferentes miembros del equipo. A menudo el director del equipo prepara el diagrama de flujo del sistema y, luego, lo divide en módulos para los miembros del equipo.

A un miembro del equipo se le puede asignar la tarea de desarrollar un archivo de macro definiciones, el cual podría contener macro definiciones que manejen las operaciones de E/S para el sistema. A otro miembro del equipo se le puede asignar la tarea de desarrollar los procedimientos utilizados por el sistema. En casi todos los casos, los procedimientos se organizan en forma de archivos de biblioteca que se ligan con los módulos del programa. Asimismo, se podrían emplear diversos archivos o módulos de programas y buena documentación.

Este método requiere una comunicación considerable entre los miembros del equipo y buena documentación. La documentación es la clave para que haya una interface correcta de todos los módulos. También la comunicación entre los miembros del equipo es una función clave en este método.

6–2 EMPLEO DEL TECLADO Y EL MONITOR DE VIDEO

En la actualidad hay pocos programas que no utilicen el teclado y el monitor de video. En esta sección se explica la forma de usar el teclado y el monitor conectados con la IBM PC o compatible que trabajen con MSDOS o PCDOS.

Lectura del teclado con funciones del DOS

El teclado de la computadora personal se lee por medio de una llamada a una función del DOS. En el apéndice A aparece una lista completa de las llamadas a las funciones del DOS. En esta sección se utiliza la INT 21H con llamadas a diversas funciones del DOS para leer el teclado. Los datos leídos en el teclado están en la forma de código ASCII o de código ASCII extendido.

Los datos en código ASCII aparecen como se muestra en la tabla 1-7 en la sección 1-7. El código extendido de la tabla 1-8 se aplica sólo a datos impresos o exhibidos y no a datos del teclado. Se debe tener en cuenta que los códigos ASCII en la tabla 1-7 corresponden a la mayor parte de las teclas del teclado. Por medio del teclado también están disponibles los datos del código ASCII extendido. En la tabla 6-1 aparece la mayor parte de los códigos ASCII extendidos obtenidos con teclas y combinaciones de ellas. Se verá que la mayor parte de las teclas en el teclado tienen inscripciones para códigos alternos. Las teclas de función tienen cuatro grupos de códigos que se seleccionan con las teclas de función, las teclas de cambio de función, las teclas de función alterna y las teclas de función de control.

Hay tres formas de leer el teclado: en la primera se lee una tecla y produce el eco (o sea exhibición) de la tecla en el monitor. En la segunda forma, sólo se prueba si una tecla ha sido oprimida y, si es así, lee la tecla; de lo contrario retorna sin nada. La tercera forma permite leer una línea completa de caracteres del teclado.

TABLA 6–1 Teclas con código ASCII extendido

Primero / Segundo	0	1	2	3	4	5	6	7	8
0	—	aQ	aD	aB	—	bajar flecha	cF3	aF9	a9
1	aESC	aW	aF	aN	—	bajar página	cF4	aF10	a0
2	—	aE	aG	aM	—	insertar	cF5	—	a-
3	c2	aR	aH	a,	—	borrar	cF6	—	a=
4	—	aT	aJ	a.	—	sF1	cF7	—	—
5	—	aY	aK	a/	—	sF2	cF8	—	F11
6	—	aU	aL	—	—	sF3	cF9		F12
7	—	aI	a;	a*	inicial	sF4	cF10	—	sF11
8	—	aO	a'	—	subir flecha	sF5	aF1	a1	sF12
9	—	aP	a`	—	subir página	sF6	aF2	a2	cF11
A	—	a[	—	a-	—	sF7	aF3	a3	cF12
B	—	a]	a\	—	flecha izquierda	sF8	aF4	a4	aF11
C	—	aENT	aZ	—	—	sF9	aF5	a5	aF12
D	—	—	aX	—	flecha derecha	sF10	aF6	a6	—
E	aBS	aA	aC	—	a+	cF1	aF7	a7	—
F	sTAB	aS	—	—	tecla de fin	cF2	aF8	a8	—

Notas: a= tecla ALT, c= tecla Control, s= tecla de cambio a mayúsculas.

Lectura de una tecla con eco. En el ejemplo 6-10 se muestra la forma en que se lee una tecla del teclado y se envía como eco al monitor. Aunque ésta es la forma más fácil para leer una tecla, también es la más limitada, porque siempre envía como eco un carácter al monitor, aunque sea indeseado. La función 01H del DOS también responde a la secuencia de tecla C y da salida a DOS si se oprime.

EJEMPLO 6–10

```
0000                TECL    PROC    FAR

0000 B4 01                  MOV     AH,1        ;función 01H
0002 CD 21                  INT     21H         ;leer tecla
0004 0A C0                  OR      AL,AL       ;probar si hay 00H
0006 75 03                  JNZ     TECL1
0008 CD 21                  INT     21H         ;obtener extendido
000A F9                     STC                 ;indicar extendido

000B                TECL1

000B CB                     RET

000C                KEY     ENDP
```

Para leer un carácter y enviarlo como eco, se carga el registro AH con el número 01H de función del DOS, seguido por la instrucción INT 21H. Al retornar de la INT 21H, el registro AL contiene el código ASCII del carácter que se tecleó y en el monitor aparece también ese carácter.

Si AL = 0 después del retorno, hay que ejecutar otra vez la instrucción INT 21H para obtener el carácter en el código ASCII extendido (véase la tabla 6-1). El procedimiento del ejemplo 6-10 retorna con el acarreo activado (1) para indicar un carácter en ASCII extendido y (0) el acarreo para indicar un carácter ASCII normal.

Lectura de una tecla sin eco. La función óptima para leer un solo carácter tecleado, es la función 06H, la cual lee una tecla pero no la envía como eco al monitor. También permite los caracteres ASCII extendidos pero no responde a control C. En esta función se utiliza AH para el número de función (06H) y a DL = 0FFH para indicar que la llamada a la función (INT 21H) leerá el teclado pero sin que haya eco.

EJEMPLO 6–11

```
0000                TECLAS  PROC    FAR

0000 B4 06                  MOV     AH,6            ;función 01H
0002 B2 FF                  MOV     DL,0FFH
0004 CD 21                  INT     21H             ;leer tecla
0006 74 F8                  JE      TECLAS          ;si no hay tecla
0008 0A C0                  OR      AL,AL           ;probar si es 00H
000A 75 03                  JNE     TECLAS1
000C CD 21                  INT     21H             ;obtener extendido
000E F9                     STC                     ;indicar extendido

000F                TECLAS1:

000F CB                     RET

0010                TECLAS  ENDP
```

En el ejemplo 6-11 se presenta un procedimiento que utiliza la función 06H para leer el teclado. Eso ocurre como en el ejemplo 6-10 excepto que no se envía ningún carácter como eco al monitor.

Si se examina el procedimiento, se encontrará otra diferencia más. La función 06H retorna de la INT 21H aunque no se haya oprimido ninguna tecla, mientras que la función 01H espera a que se oprima una tecla. Se trata de una diferencia importante que se debe tener en cuenta. Esta característica permite que un programa ejecute otras tareas entre cada verificación del teclado para determinar si se oprimió una tecla.

Leer una línea completa con eco. En ocasiones es ventajoso leer una línea completa de datos con una sola llamada a función. La función número 0AH lee una línea completa de información, hasta de 255 caracteres, del teclado. Continúa con la adquisición de datos del teclado hasta que, ya sea, que oprima la tecla de entrada (0DH) o bien se termine el conteo de caracteres. Esta función requiere que AH = 0AH y que DS:DX direccione la zona de memoria del teclado (buffer) es (una zona de memoria en donde se almacenan los datos en ASCII). El primer byte de esa zona de la memoria debe contener el número máximo de caracteres leídos con esa función. si se excede del número máximo, la función retorna como si se hubiera oprimido la tecla de entrada. El segundo byte de esa zona de la memoria contiene la cuanta real del número de caracteres tecleados y las localidades restantes en la memoria intermedia contienen los datos de ASCII del teclado.

En el ejemplo 6-12 se muestra la forma en que esta función lee 2 líneas de información en dos zonas de la memoria (BUF1 y BUF2). Antes de llamar a la función del DOS con el procedimiento

LINEA, se carga el primer byte de la memoria con un 255, de modo de poder teclear hasta 255 caracteres. Si se ensambla y se ejecuta este programa, se aceptará la primera línea y lo mismo ocurrirá con la segunda. El único problema es que la segunda línea aparece encima de la primera. En la siguiente sección se explica la forma de dar salida a caracteres hacia el monitor para resolver este problema.

EJEMPLO 6–12

```
0000                     COD    SEGMENT

                                ASSUME CS:COD

0000 0101[                      BUF1   DB   257 DUP (?)
           ????
               ]
0102 0101[                      BUF2   DB   257 DUP (?)
           ????
               ]

0204                     MAIN   PROC   FAR

0204 8C C8                      MOV    AX,CS
0206 8E D8                      MOV    DS,AX
0208 BA 0000 R                  MOV    DX,OFFSET BUF1     ;direccionar buffer 1
020B C7 06 0000 R 00FF          MOV    BUF1,255           ;conteo máximo
0211 E8 0221 R                  CALL   LINEA              ;leer primera linea

0214 BA 0102 R                  MOV    DX,OFFSET BUF2     ;direccionar buffer 2
0217 C7 06 0102 R 00FF          MOV    BUF2,255           ;conteo máximo
021D E8 0221 R                  CALL   LINEA              ;leer segunda linea

0220 CB                         RET

0221                     MAIN   ENDP

0221                     LINEA  PROC   NEAR

0221 B4 0A                      MOV    AH,0AH             ;función 0AH
0223 CD 21                      INT    21H
0225 C3                         RET

0226                     LINEA  ENDP

0226                     COD    ENDS

                                END    MAIN
```

Escritura en el monitor con funciones del DOS

Con casi cualquier programa que se escriba, los datos se deben exhibir en el monitor. Esos datos se exhiben en numerosas formas con las llamadas a funciones del DOS. Se utilizan las funciones 02H o 06H para exhibir un carácter cada vez o la función 09H para exhibir una cadena completa de caracteres. Debido a que las funciones 02H y 06H son idénticas se suele utilizar la función 06H porque también se utiliza para leer una tecla.

Exhibición de un carácter ASCII. Las funciones del 02H y 06H DOS se explican juntas porque son idénticas para exhibir datos ASCII. En el ejemplo 6-13 se ilustra la forma en que se utiliza esta

función para exhibir un retorno de carro (0DH) y una alimentación de línea (0AH). En este caso se emplea un macro llamado EXHIBE para exhibir el retorno del carro y el avance o alimentación de una línea. La combinación del retorno del carro y alimentación de línea mueve al cursor a la siguiente línea en el margen izquierdo de la pantalla del monitor. Este proceso de dos pasos se emplea para corregir el problema que ocurrió entre las líneas tecleadas en el ejemplo 6-12.

EJEMPLO 6–13

```
                     DESP     MACRO  A                     ;exhibir A

                              MOV    AH,06H
                              MOV    DL,A
                              INT    21H
                              ENDM

                              DESP   0DH                   ;retorno del carro

0000 B4 06        1           MOV    AH,06H
0002 B2 0D        1           MOV    DL,0DH
0004 CD 21        1           INT    21H

                              DESP   0AH                   ;alimentación de línea

0006 B4 06        1           MOV    AH,06H
0008 B2 0A        1           MOV    DL,0AH
000A CD 21        1           INT    21H
```

Exhibir una cadena de caracteres. Una cadena de caracteres es una serie de caracteres en código ASCII que terminan con $ (24H) cuando se emplea la función 09H del DOS. En el ejemplo 6-14 se ilustra la forma en que se exhibe un mensaje en la posición en que se encuentre el cursor en ese momento en la pantalla del monitor. La llamada 09H de función requiere que DS:DX direccione la cadena de caracteres antes de ejecutar la INT 21H.

El programa del ejemplo se puede ensamblar, ligar y ejecutarlo para producir la leyenda. (Esta es una línea de prueba) en la pantalla. Se debe tener en cuenta que al final hay una llamada adicional a otra función del DOS añadida a este programa. La función 4CH regresa al sistema al indicador de DOS al final del programa. A menudo se utiliza la función 4CH para terminar un programa y retornar a DOS. (Se debe tener en cuenta que con la función 4CH retorna con el registro AL con 00H para indicar que no hay error.)

EJEMPLO 6–14

```
0000                    COD     SEGMENT

                                ASSUME CS:COD,DS:COD

0000 0D 0A 0A 54 68 69  MES     DB     0DAH,0AH,0AH,'Esta es una línea de prueba','$'
     73 20 69 73 20 61
     20 74 65 73 74 20
     6C 69 6E 65 24

0017                    MAIN    PROC   FAR

0017 8C C8                      MOV    AX,CS                ;cargar DS
0019 8E D8                      MOV    DS,AX
```

```
001B B4 09                    MOV   AH,09H            ;función 09H
001D BA 0000 R                MOV   DX,OFFSET MES
0020 CD 21                    INT   21H

0022 B4 4C                    MOV   AH,4CH            ;función 4CH
0024 CD 21                    INT   21H

0026                 MAIN     ENDP


0026                 COD      ENDS

                              END   MAIN
```

Procedimientos de biblioteca para leer teclas con eco. En el ejemplo 6-15 se ilustran dos procedimientos que se pueden ensamblar, ligar y agregar a un archivo de biblioteca. El procedimiento LEE lee un carácter del teclado y retorna con un carácter ASCII normal o extendido en AL. Si se activa el acarreo, AL contiene un código ASCII extendido y si se desactiva el acarreo, contiene un código ASCII normal. El procedimiento ECO exhibe el carácter ASCII cargado en AL, en la posición en que esté el cursor en ese momento.

EJEMPLO 6–15

```
0000            LIB     SEGMENT

                        ASSUME CS:LIB

                        PUBLIC READ

                        PUBLIC ECHO

                ;
                ;procedimiento que lee una tecla del teclado (sin eco)
                ;si CF = 0, AL = carácter ASCII normal
                ;si CF = 1, AL = carácter ASCII extendido
                ;
0000            LEE        PROC  FAR

0000 52                    PUSH  DX               ;salvar DX
0001 B4 06                 MOV   AH,6             ;función 06H del DOS
0003 B2 FF                 MOV   DL,0FFH

0005            LEE1:

0005 CD 21                 INT   21H
0007 74 FC                 JE    LEE1             ;si no hay tecla
0009 0A C0                 OR    AL,AL            ;probar si es extendido
000B 75 03                 JNZ   LEE2             ;si es ASCII normal

000D CD 21                 INT   21H              ;obtener extendido
000F F9                    STC                    ;desactivar acarreo

0010            LEE2:

0010 5A                    POP   DX
0011 CB                    RET

0012            LEE        ENDP
```

```
                         ;procedimiento que exhibe el carácter ASCII en AL
                         ;
0012                     ECO       PROC    FAR

0012 52                            PUSH    DX
0013 B4 06                         MOV     AH,6          ;función 06H del DOS
0015 8A D0                         MOV     DL,AL         ;AL a DL
0017 CD 21                         INT     21H
0019 5A                            POP     DX
001A CB                            RET

001B                     ECO       ENDP

001B                     LIB       ENDS

                                   END
```

Empleo de llamadas de funciones de video o del BIOS

Además de las llamadas a funciones del DOS con la INT 21H, se tienen también llamadas a funciones del sistema *básico de entrada y salida* BIOS que llaman a la INT 10H. Las llamadas a la función DOS permiten leer una tecla y exhibir un carácter con facilidad pero es difícil "posicionar" el cursor en el lugar deseado en la pantalla. Las llamadas a las funciones del BIOS, permiten mayor control sobre la exhibición en la pantalla que las llamadas a funciones del DOS. Las llamadas a las funciones del BIOS también requieren menos tiempo de ejecución que las llamadas a las funciones del DOS.

Posición del cursor. Antes de colocar cualquier información en la pantalla, se debe conocer la posición del cursor. Esto permite borrar la pantalla y volver a empezar en cualquier posición que se desee. 02H permite colocar el cursor en cualquier posición en la pantalla. En la tabla 6-2 se muestra el contenido de diversos registros para las funciones 02H y 03H.

El número de página, en el registro BH debe ser de cero antes de inicializar la posición del cursor. La mayor parte de los programas, no acceden por lo general a las otras páginas (1 a 7) de la exhibición en la pantalla. El número de página, a menudo, se echa al olvido después de una lectura del cursor. La página 0 está disponible en los modos de funcionamiento texto CGA (*adaptador gráfico a color*), EGA (*adaptador mejor a color*) y VGA (*arreglo gráfico de vídeo*).

Para la posición del cursor se supone que la columna izquierda en la página es la columna 0 y avanza a lo largo de la línea hasta la columna 79. El renglón 0 es el más alto mientras que el 23 es el último renglón en la pantalla. Con ello se supone que el modo de texto seleccionado para el adaptador de vídeo es de 80 caracteres por renglón 24 renglones. Hay otros modos texto disponibles como 40 × 24 y 96 × 43.

TABLA 6–2 Funciones de la INT 10H del BIOS

AH	*Descripción*	*Parámetros*
02H	Inicializa la posición del cursor	DH = Renglón DL = Columna BH = No. de página
03H	Lee posición del cursor	BH = No. de página DH = Renglón DL = Columna

EJEMPLO 6–16

```
0000                    CODE    SEGMENT

                                ASSUME CS:CODE

0000                    MAIN    PROC    FAR

                        CASA    MACRO

                                MOV     AH,2            ;;inicializa la posición del cursor
                                MOV     BH,0            ;;página 0
                                MOV     DX,0            ;;renglón 0, columna 0
                                INT     10H

                                ENDM
                                CASA
0000 B4 02        1             MOV     AH,2
0002 B7 00        1             MOV     BH,0
0004 BA 0000      1             MOV     DX,0
0007 CD 10        1             INT     10H

0009 B9 0780                    MOV     CX,1920
000C B4 06                      MOV     AH,6
000E B2 20                      MOV     DL,''           ;espacio

0010                    PRINC1:

0010 CD 21                      INT     21H             ;exhibir espacio
0012 E2 FC                      LOOP    MAIN1           ;repetir 1920 veces

                                HOME
0014 B4 02        1             MOV     AH,2
0016 B7 00        1             MOV     BH,0
0018 BA 0000      1             MOV     DX,0
001B CD 10        1             INT     10H

001D B4 4C                      MOV     AH,4CH          ;salir a DOS
001F CD 21                      INT     21H

0021                    MAIN    ENDP

0021                    CODE    ENDS

                                END     MAIN
```

En el ejemplo 6-16 se ilustra la forma en que se utiliza la llamada a la función 02H de la INT 10H del BIOS para borrar la pantalla del monitor. Este es sólo un método para borrar la pantalla. Se debe tener en cuenta que la primera llamada de función ubica al cursor en el renglón 0 y columna 0, en lo que se llama "posición de *casa*". Después, se utiliza la llamada a una función 6 del DOS para escribir 1920 (80 caracteres por línea × 24 líneas,) como espacios en blanco (20H) en la pantalla y para volver el cursos a su posición inicial.

Si este ejemplo se ensambla, liga y ejecuta, surge un problema. Este programa es demasiado lento para que sea útil en la mayor parte de los casos. Para corregir este problema, se emplea otra llamada a una función del BIOS. Luego, se puede usar la función de desplazamiento ("scroll") de pantalla (06H) para limpiar la pantalla a mucha mayor velocidad.

EJEMPLO 6–17

```
0000                CODE   SEGMENT
                           ASSUME CS:CODE
0000                MAIN   PROC   FAR
                    CASA   MACRO
                           MOV    AH,2            ;;inicializar posición del cursor
                           MOV    BH,0            ;;página 0
                           MOV    DX,0            ;;renglón 0, columna 0
                           INT    10H
                           ENDM
0000 32 FF                 XOR    BH,BH           ;página 0
0002 B4 08                 MOV    AH,8            ;leer atributos
0004 CD 10                 INT    10H
0006 8A DF                 MOV    BL,BH
0008 8A FC                 MOV    BH,AH
000A 2B C9                 SUB    CX,CX
000C BA 184F               MOV    DX,184FH
000F B8 0600               MOV    AX,0600H        ;borrar página 0
0012 CD 10                 INT    10H
                           CASA
0014 B4 02       1         MOV    AH,2
0016 B7 00       1         MOV    BH,0
0018 BA 0000     1         MOV    DX,0
001B CD 10       1         INT    10H
001D B4 4C                 MOV    AH,4CH          ;salir a DOS
001F CD 21                 INT    21H
0021                MAIN   ENDP
0021                CODE   ENDS
                           END    MAIN
```

La función 06H se emplea con un 00H en AL para borrar toda la pantalla. Esto permite volver a escribir el ejemplo 6-16 de modo que la pantalla se borre a una velocidad mucho más alta. En el ejemplo 6-17 aparece un programa mejor para borrar la pantalla y poner el cursor en su posición inicial. En este caso, la llamada a la función 08H, lee los atributos de los caracteres para borrar la pantalla. Luego, se los carga en los registros correctos y se carga DX con el tamaño de la pantalla, 4FH (79) y 18H (24). Si este programa se ensambla, liga, ejecuta y compara con el ejemplo 6-16, hay una gran diferencia en la velocidad a la cual se borra la pantalla. Consulte en el apéndice A otras llamadas a la función INT 10H del BIOS que pueden ser útiles para la aplicación que se desea. Además, en el apéndice A aparece una lista completa de todas las funciones de interrupciones INT disponibles para casi todas las computadoras.

6–3 CONVERSIONES DE DATOS

En los sistemas de computadora, los datos rara vez están en la forma correcta. Una tarea principal del sistema es convertir los datos de una forma a otra. En esta sección se describen las conversiones entre binario y ASCII. Los datos binarios se extraen de un registro o de la memoria y se convierten en ASCII para exhibirlos en la pantalla. Los datos en ASCII se convierten a binarios al escribirlos en el teclado. También se explica la conversión entre datos ASCII y hexadecimales.

Conversión de binario a ASCII

La conversión de binario a ASCII se logra en dos formas. (1) con la instrucción AAM si el número es menor de 100 o (2) con una serie de divisiones decimales (dividir entre 10). Ambas técnicas se presentan en esta sección.

La instrucción AAM convierte el valor que hay en AX a un número de dos dígitos en BCD no empacado. Si el número en AX es 0062H (98 decimal) antes de ejecutar AAM, AX contendrá un 0908H después de que se ejecute AAM. No trata de un código ASCII, sino que se convierte a ASCII al agregar un 3030H a AX. En el ejemplo 6-18 se ilustra un procedimiento que procesa el valor binario en AL (0 a 99) y lo exhibe en la pantalla como decimal. Este procedimiento borra un cero precedente, lo cual ocurre para los números 0 a 9 con un código especial ASCII.

EJEMPLO 6–18

```
0000                DESP    PROC    FAR

0000 52                     PUSH    DX              ;salvar DX
0001 32 E4                  XOR     AH,AH           ;borrar AH
0003 D4 0A                  AAM                     ;convertir a BCD
0005 80 C4 20               ADD     AH,20H          ;sumar 20H
0008 80 FC 20               CMP     AH,20H          ;probar si hay un cero avanzado
000B 74 03                  JE      DESP1           ;si hay cero inicial
000D 80 C4 10               ADD     AH,10H          ;convertir a ASCII

0010                DESP1:

0010 50                     PUSH    AX
0011 8A D4                  MOV     DL,AH           ;exhibir primer digito
0013 B4 06                  MOV     AH,6
0015 CD 21                  INT     21H
0017 58                     POP     AX
0018 04 30                  ADD     AL,30H          ;convertir a ASCII
001A 8A D0                  MOV     DL,AL
001C B4 06                  MOV     AH,6            ;exhibir segundo digito
001E CD 21                  INT     21H
0020 5A                     POP     DX              ;recuperar DX
0021 CB                     RET

0022                DESP    ENDP
```

La razón por la cual AAM convierte cualquier número entre 0 y 99 a un número BCD sin empacar, de dos dígitos, es porque divide AX entre 10. El resultado se deja en AX, con lo cual AH contiene el cociente y AL el residuo. Este mismo sistema de dividir entre 10 se puede ampliar o

expandir para convertir cualquier número completo de binario a una cadena de caracteres ASCII que se puede exhibir en la pantalla. El algoritmo para convertir de binario a ASCII es:

1. Dividir entre 10 y conservar el cociente en la pila como dígito BCD significativo.
2. Repetir el paso 1 hasta que el cociente sea un 0.
3. Recuperar cada residuo y sumarle un 30H para convertir a ASCII antes de exhibir o imprimir.

EJEMPLO 6–19

```
0000              DESPX   PROC   FAR

0000 52                   PUSH   DX        ;salvar BX, CX y DX
0001 51                   PUSH   CX
0002 53                   PUSH   BX

0003 33 C9                XOR    CX,CX     ;borrar CX
0005 BB 000A              MOV    BX,10     ;cargar 10

0008              DESPX1:

0008 33 D2                XOR    DX,DX     ;borrar DX
000A F7 F3                DIV    BX
000C 52                   PUSH   DX        ;salvar el residuo
000D 41                   INC    CX        ;contador del residuo
000E 0B C0                OR     AX,AX     ;probar el cociente
0010 75 F6                JNZ    DESPX1    ;si no es cero

0012              DESPX2:

0012 5A                   POP    DX        ;exhibir número
0013 B4 06                MOV    AH,6
0015 80 C2 30             ADD    DL,30H    ;convertir a ASCII
0018 CD 21                INT    21H
001A E2 F6                LOOP   DESPX2    ;repetir

001C 5B                   POP    BX        ;recuperar BX, CX y DX
001D 59                   POP    CX
001E 5A                   POP    DX
001F CB                   RET

0020              DESPX   ENDP
```

En el ejemplo 6-19 se muestra la forma en que el contenido de 16 bits sin signo de AX se convierte a ASCII y se exhibe en la pantalla. En este caso se divide entre 10 y se salva el residuo en la pila después de cada división, para su posterior conversión a ASCII. Después de haber convertido todos los dígitos, se extraen los residuos de la pila y se los convierte a código ASCII para exhibir el resultado en la pantalla. Este procedimiento también borra cualesquiera ceros precedentes que pudieren ocurrir.

Conversión de ASCII a binario

Las conversiones de ASCII a binario suelen comenzar con una entrada del teclado. Si se oprime una sola tecla, la conversión ocurre cuando se resta un 30H del número. Si se oprime más de una tecla, la conversión de ASCII a binario todavía requiere restar 30H, pero hay un paso adicional.

Después de restar 30H, el número se suma al resultado una vez que el resultado anterior se multiplicó por 10. El algoritmo para convertir de ASCII a binario es:

1. Empezar con un resultado binario de 0.
2. Restar 30H del carácter tecleado en el teclado para convertirlo a BCD.
3. Multiplicar el resultado por 10 y sumar el nuevo dígito en BCD.
4. Repetir los pasos 2 y 3 hasta que el carácter tecleado no sea un número en código ASCII.

EJEMPLO 6–20

```
0000              LEEN    PROC   FAR

0000 53                   PUSH   BX           ;salvar BX y CX
0001 51                   PUSH   CX
0002 B9 000A              MOV    CX,10        ;cargar 10
0005 33 DB                XOR    BX,BX        ;borrar el resultado

0007              LEEN1:
0007 B4 06                MOV    AH,6         ;leer tecla
0009 B2 FF                MOV    DL,0FFH
000B CD 21                INT    21H
000D 74 F8                JE     LEEN1        ;esperar a la tecla

000F 3C 30                CMP    AL,'0'       ;probar contra de 0
0011 72 18                JB     LEEN2        ;si es menor de 0
0013 3C 39                CMP    AL,'9'       ;probar contra de 9
0015 77 14                JA     LEEN2        ;si es mayor de 9

0017 8A D0                MOV    DL,AL        ;eco tecla
0019 CD 21                INT    21H

001B 2C 30                SUB    AL,'0'       ;convertir a BCD

001D 50                   PUSH   AX
001E 8B C3                MOV    AX,BX        ;multiplicar por 10
0020 F7 E1                MUL    CX
0022 8B D8                MOV    BX,AX        ;salvar el producto
0024 58                   POP    AX
0025 32 E4                XOR    AH,AH
0027 03 D8                ADD    BX,AX        ;sumar BCD al producto
0029 EB DC                JMP    LEEN1        ;repetir

002B              LEEN2:

002B 8B C3                MOV    AX,BX        ;cargar binario en AX
002D 59                   POP    CX           ;recuperar BX y CX
002E 5B                   POP    BX
002F CB                   RET

0030              LEEN    ENDP
```

En el ejemplo 6-20 se ilustra el procedimiento para poner en ejecución este algoritmo. En este caso, el número binario retorna al registro AX como un resultado de 16 bits. Si se requiere un resultado más grande, hay que volver a efectuar el procedimiento para una suma de 32 bits. Cada vez que se llama a este procedimiento, lee un número en el teclado hasta que se oprime una tecla que no sea del 0 al 9.

Exhibición y lectura de datos hexadecimales

Los datos hexadecimales son más fáciles de leer desde el teclado y en la pantalla, que los datos decimales. Este tipo de datos no se utilizan en el nivel de aplicaciones sino de sistema. Los datos del sistema a menudo son hexadecimales y se deben exhibir como tales o bien leerlos del teclado como datos hexadecimales.

Lectura de datos hexadecimales. Los datos hexadecimales aparecen como 0 a 9 y como A hasta F. Los códigos ASCII obtenidos con el teclado para datos hexadecimales son 30H hasta 39H para los números 0 hasta 9; 41H a 46H (A hasta F) o 61H a 66H (a hasta f) para las letras. Para que sea útil un procedimiento que lea los datos hexadecimales, debe aceptar letras minúsculas y mayúsculas.

EJEMPLO 6–21

```
0000                LEEH    PROC   NEAR

0000 3C 39                  CMP    AL,'9'
0002 76 08                  JBE    CONV2          ;si es un número
0004 3C 61                  CMP    AL,'a'
0006 72 02                  JB     CONV1          ;si son letras mayúsculas
0008 2C 20                  SUB    AL,20H

000A                LEEH1:

000A 2C 07                  SUB    AL,7

000C                LEEH2:

000C 2C 30                  SUB    AL,'0'
000E C3                     RET

000F                LEEH    ENDP

000F                LEEH    PROC   FAR

000F 51                     PUSH   CX             ;salvar BX y CX
0010 53                     PUSH   BX
0011 B9 0004                MOV    CX,4           ;cargar contador de corrimiento
0014 8B F1                  MOV    SI,CX          ;cargar conteo
0016 33 DB                  XOR    BX,BX          ;borrar el resultado

0018                LEEH1:

0018 B4 06                  MOV    AH,6           ;leer tecla
001A B2 FF                  MOV    DL,0FFH
001C CD 21                  INT    21H
001E 74 F8                  JE     LEEH1          ;esperar a la tecla
0020 8A D0                  MOV    DL,AL          ;eco
0022 CD 21                  INT    21H
0024 E8 0000 R              CALL   CONV           ;convertir a hexadecimal
0027 D3 E3                  SHL    BX,CL          ;recorrer el resultado
0029 02 D8                  ADD    BL,AL          ;sumar AL al resultado
002B 4E                     DEC    SI
002C 75 EA                  JNZ    LEEH1          ;repetir 4 veces
```

```
002E 5B              POP   BX        ;recuperar BX y CX
002F 59              POP   CX
0030 CB              RET

0031          READH  ENDP
```

En el ejemplo 6-21 se muestran dos procedimientos: uno convierte el contenido de datos en AX de código ASCII a un solo dígito hexadecimal, mientras que el segundo lee un número hexadecimal de 4 números del teclado y lo retorna al registro AX. El procedimiento se puede modificar para leer cualquier número hexadecimal en el teclado.

Exhibición de datos hexadecimales. Para exhibir datos hexadecimales, un número se debe dividir en segmentos de 4 bits que se convierten a dígitos hexadecimales. Para lograr la conversión, se suma un 30H a los números 0 hasta 9 y un 37H a las letras A hasta F.

EJEMPLO 6–22

```
0000          DESPH   PROC  FAR

0000 51               PUSH  CX        ;salvar CX y DX
0001 52               PUSH  DX
0002 B1 04            MOV   CL,4      ;cargar contador de rotacion
0004 B5 04            MOV   CH,4      ;cargar contador de digitos

0006          DESPH1:

0006 D3 C0            ROL   AX,CL     ;colocar el número
0008 50               PUSH  AX        ;salvarlo
0009 24 0F            AND   AL,0FH    ;obtener digito hexadecimal
000B 04 30            ADD   AL,30H    ;ajustarlo
000D 3C 39            CMP   AL,'9'    ;probar contra de 9
000F 76 02            JBE   DESPH2    ;si es de 0 a 9
0011 04 07            ADD   AL,7      ;ajustarlo

0013          DESPH2:

0013 B4 06            MOV   AH,6
0015 8A D0            MOV   DL,AL
0017 CD 21            INT   21H       ;exhibir digito
0019 58               POP   AX        ;recuperar AX
001A FE CD            DEC   CH
001C 75 E8            JNZ   DESPH2    ;repetir

001E 5A               POP   DX        ;restaurar CX y DX
001F 59               POP   CX
0020 CB               RET

0021          DESPH   ENDP
```

En el ejemplo 6-22 se presenta un procedimiento que exhibe el contenido del registro AX en la pantalla. En este caso, el número se hace girar a la izquierda para que el dígito que está más a la izquierda, sea el primero que se exhibe. Debido a que AX contiene un número hexadecimal de 4 dígitos, el procedimiento exhibe 4 dígitos hexadecimales.

Empleo de tablas para conversiones de datos

Las tablas se utilizan a menudo para convertir datos de una forma a otra. Una tabla está formada en la memoria como una lista de datos que se consultan por un procedimiento para efectuar conversiones. En el caso de muchas tablas, a menudo se puede utilizar la instrucción XLAT para buscar o consultar datos en una tabla , siempre y cuando la tabla contenga datos de 8 bits de ancho y con longitud igual o menor de 256 bytes.

Conversión de BCD a un código de 7 segmentos. Una aplicación sencilla en la que se emplea una tabla es la conversión de BCD al código de 7 segmentos. En el ejemplo 6-23 se ilustra una tabla que contiene los códigos de 7 segmentos para los números 0 a 9. Estos códigos se emplean con la exhibición visual de 7 segmentos que se muestran en la figura 6-1. En esta exhibición de 7 segmentos se utilizan entradas activas en alto (1 lógico) para encender un segmento. El código está dispuesto de modo que el segmento *a* esté en la posición de bit 0 y el segmento *g* esté en la posición de bit 6. La posición de bit 7 es cero en este ejemplo, pero se puede emplear para exhibir un punto decimal.

EJEMPLO 6–23

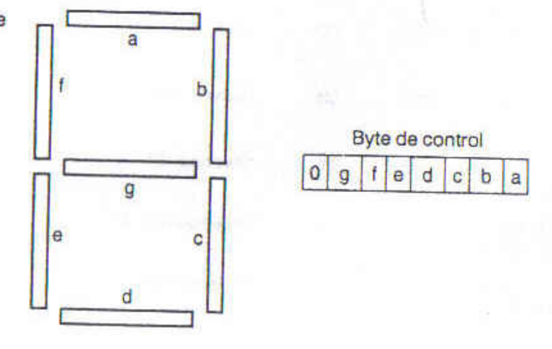
```
0000                SEG7    PROC    FAR

0000 53                     PUSH    BX
0001 BB 0008 R              MOV     BX,OFFSET TABLA
0004 2E: D7                 XLAT    CS:TABLA            ;véase el texto
0006 5B                     POP     BX
0007 CB                     RET

0008 3F             TABLA   DB      3FH                 ;0
0009 06                     DB      6                   ;1
000A 5B                     DB      5BH                 ;2
000B 4F                     DB      4FH                 ;3
000C 66                     DB      66H                 ;4
000D 6D                     DB      6DH                 ;5
000E 7D                     DB      7DH                 ;6
000F 07                     DB      7                   ;7
0010 7F                     DB      7FH                 ;8
0011 6F                     DB      6FH                 ;9

0012                SEG7    ENDP
```

FIGURA 6–1 Exhibición visual de 7 segmentos.



El procedimiento que efectúa la conversión contiene sólo dos instrucciones y se supone que AL contiene el dígito que se va a convertir al código de 7 segmentos. Una de las instrucciones direcciona a la tabla porque carga su dirección en BX y la otra efectúa la conversión y retorna el código de 7 segmentos en AL.

Debido a que la tabla de consulta está ubicada en el segmento de código y a que la instrucción XLAT accede al segmento de datos en forma implícita la instrucción XLAT incluye un cambio del segmento. Se debe tener en cuenta que se agrega un operando mudo (TABLA) a la instrucción XLAT para poder agregar el prefijo (CS:) de cambio al segmento de código para la instrucción. Las instrucciones LODS y MOVS también se cambian de segmento en la misma forma que XLAT con el empleo de un operando ficticio.

Empleo de una tabla para accesar a datos ASCII. Algunas técnicas para programación requieren convertir los códigos numéricos a cadenas de caracteres en ASCII. Por ejemplo, supóngase que se necesita exhibir los días de la semana para un programa de calendario. Debido a que la cantidad de caracteres ASCII para cada día es diferente, se debe emplear algún tipo de tabla de los días de la semana que están en código ASCII.

EJEMPLO 6–24

```
0000                    DIAS    PROC    FAR

0000 52                         PUSH    DX              ;salvar DX y SI
0001 56                         PUSH    SI
0002 BE 001B R                  MOV     SI,OFFSET TABD  ;direccionar TABD
0005 32 E4                      XOR     AH,AH           ;borrar AX
0007 03 C0                      ADD     AX,AX           ;duplicar AX
0009 03 F0                      ADD     SI,AX           ;modificar dirección de la tabla
000B 2E: 8B 14                  MOV     DX,CS:[SI]      ;obtener dirección de la cadena
000E 8C C8                      MOV     AX,CS           ;cambiar segmento de datos
0010 1E                         PUSH    DS
0011 8E D8                      MOV     DS,AX
0013 B4 09                      MOV     AH,9
0015 CD 21                      INT     21H             ;exhibir la cadena
0017 1F                         POP     DS
0018 5E                         POP     SI              ;recuperar DX y SI
0019 5A                         POP     DX
001A CB                         RET

001B 0029 R 0031 R      DTAB    DW      DOM,LUN,MAR,MIE,JUE,VIER,SAB
     0039 R 0042 R
     004D R 0057 R
     005F R

0029 53 75 6E 64 61     SUN     DB      'Domingo $'
     79 20 24

0031 4D 6F 6E 64 61     MON     DB      'Lunes $'
     79 20 24

0039 54 75 65 73 64     TUE     DB      'Martes $'
     61 79 20 24

0042 57 65 64 6E 65     WED     DB      'Miércoles $'
     73 64 61 79 20 24

004D 54 68 75 72 73     THU     DB      'Jueves $'
     64 61 79 20 24
```

```
0057 46 72 69 64 61    FRI     DB      'Viernes $'
     79 20 24

005F 53 61 74 75 72    SAT     DB      'Sábado $'
     64 61 79 20 24

0069                   DIAS    ENDP
```

En el ejemplo 6-24 se presenta una tabla de consulta para cadenas de caracteres ASCII en el segmento de código. Cada cadena de caracteres contiene un día de la semana en código ASCII. La tabla señala cada día de la semana. El procedimiento que accesa a cada día de la semana utiliza el registro AL y los números 0 a 6 significan domingo hasta sábado. Si AL contiene un 2 cuando se llama este procedimiento, entonces aparece en la pantalla la palabra *Martes*.

Para que este procedimiento direccione a la tabla, primero hay que cargar su dirección en el registro SI. Después el número que hay en AL se convierte a un número de 16 bits y se duplica porque la tabla contiene 2 bytes para cada entrada. Luego, este índice se agrega a SI para direccionar la entrada correcta en la tabla de consulta. Ahora, la dirección de la cadena de caracteres en ASCII se carga en DX por medio de la instrucción MOV DX,CS:[SI].

Antes de llamar la instrucción INT 21H DOS, se salva el registro DS en la pila y se carga con la dirección del segmento CS. Esto permite que la función 09H del DOS (exhibir una cadena) se utilice para exhibir el día de la semana. Este procedimiento convierte los números 0 a 6 a los días de la semana.

Un programa de ejemplo con el empleo de conversiones de datos

Se requiere un programa de ejemplo para combinar algunas de las funciones del DOS para conversión de datos. Supóngase que se debe exhibir la hora y la fecha en la pantalla. Este programa de ejemplo (véase ejemplo 6-25) exhibe la hora como 10:45 P.M. y la fecha como martes, mayo 14 de 1991. El programa es corto porque llama a un procedimiento que exhibe la hora y a otro que exhibe la fecha.

EJEMPLO 6–25 (página 1 de 5)

```
0000                        STAC    SEGMENT STACK

0000 0100[                          DW      256 DUP (?)         ;inicializar la pila
                ????
                    ]

0200                        STAC    ENDS

0000                        DAT     SEGMENT

0000 0026 R 002F R 0038 R   DIAS    DW    DOM,LUN,MAR,MIE,JUE,VIER,SAB
     0042 R 004E R 0059 R
     0062 R
000E 006D R 0076 R 0080 R   MES     DW    ENE,FEB,MAR,ABR,MAY,JUN,JUL,AGO,SEP,OCT,NOV,DIC
     0087 R 008E R 0093 R
     0099 R 009F R 00A7 R
     00B2 R 00BB R 00C5 R
```

EJEMPLO 6–25 (página 2 de 5)

```
0026 53 75 6E 64 61 79          SUN    DB    'Domingo, $'
     2C 20 24
002F 4D 6F 6E 64 61 79          MON    DB    'Lunes, $'
     2C 20 24
0038 54 75 65 73 64 61          TUE    DB    'Martes, $'
     79 2C 20 24
0042 57 65 64 6E 65 73          WE     DB    'Miércoles, $'
     64 61 79 2C 20 24
004E 54 68 75 72 73 64          THU    DB    'Jueves, $'
     61 79 2C 20 24
0059 46 72 69 64 61 79          FRI    DB    'Viernes, $'
     2C 20 24
0062 53 61 74 75 72 64          SAT    DB    'Sábado, $'
     61 79 2C 20 24

006D 4A 61 6E 75 61 72          JAN    DB    'Enero, $'
     79 20 24
0076 46 65 62 72 75 61          FEB    DB    'Febrero $'
     72 79 20 24
0080 4D 61 72 63 68 20          MAR    DB    'Marzo $'
     24
0087 41 70 72 69 6C 20          APR    DB    'Abril $ '
     24
008E 4D 61 79 20 24             MAY    DB    'Mayo $'
0093 4A 75 6E 65 20 24          JUN    DB    'Junio $'
0099 4A 75 6C 79 20 24          JUL    DB    'Julio $'
009F 41 75 67 75 73 74          AUG    DB    'Agosto $'
     20 24
00A7 53 65 70 74 65 6D          SEP    DB    'Septiembre $'
     62 65 72 20 24
00B2 4F 63 74 6F 62 65          OCT    DB    'Octubre $'
     72 20 24
00BB 4E 6F 76 65 6D 62          NOV    DB    'Noviembre $'
     65 72 20 24
00C5 44 65 63 65 6D 62          DC     DB    'Diciembre $'
     65 72 20 24

00CF 0D 0A 24                   CRLF   DB    13,10,'$'
00D2 2E 4D 2E 20 20 24          MES1   DB    '.M. $'
00D8 2C 20 31 39 24             MES2   DB    ',19$'
00DD 2C 20 32 30 24             MES3   DB    ',20$'

00E2                     DAT    ENDS

0000                     COD    SEGMENT

                                ASSUME CS:COD,DS:DAT,SS:STAC

0000                     PRAL   PROC   FAR              ;programa principal

0000 B8 -- R                    MOV    AX,DAT           ;cargar DS
0003 8E D8                      MOV    DS,AX

0005 BA 00CF R                  MOV    DX,OFFSET CRLF   ;obtener nueva línea
0008 B4 09                      MOV    AH,9
000A CD 21                      INT    21H

000C E8 001D R                  CALL   HORA             ;exhibir la hora
000F E8 007E R                  CALL   FECHA            ;exhibir la fecha
```

EJEMPLO 6–25 (página 3 de 5)

```
0012 BA 00CF R              MOV    DX,OFFSET CRLF       ;obtener nueva linea
0015 B4 09                  MOV    AH,9
0017 CD 21                  INT    21H

0019 B4 4C                  MOV    AH,4CH               ;salir a DOS
001B CD 21                  INT    21H

001D              PRAL      ENDP

001D              HORA      PROC   NEAR                 ;exhibir hora, XX:XX AM

001D B4 2C                  MOV    AH,2CH               ;obtener línea
001F CD 21                  INT    21H

0021 B7 41                  MOV    BH,'A'               ;inicializar hora AM

0023 80 FD 0C               CMP    CH,12                ;probar contra 12
0026 72 05                  JB     HORA1                ;si son AM

0028 B7 50                  MOV    BH,'P'               ;inicializar hora PM
002A 80 ED 0C               SUB    CH,12                ;poner en hora
002D              HORA1:

002D 0A ED                  OR     CH,CH                ;probar si son 0 horas
002F 75 02                  JNE    HORA2                ;si no son 0 horas
0031 B5 0C                  MOV    CH,12                ;ponerlo en las 12 horas
0033              HORA2:

0033 8A C5                  MOV    AL,CH                ;obtener las horas
0035 32 E4                  XOR    AH,AH                ;borrar AH
0037 D4 0A                  AAM                         ;convertir a BCD
0039 0A E4                  OR     AH,AH                ;probar décimas de hora
003B 74 09                  JZ     HORA3                ;si no hay décimas de hora
003D 50                     PUSH   AX
003E 8A C4                  MOV    AL,AH
0040 04 30                  ADD    AL,'0'               ;convertir a ASCII
0042 E8 0075 R              CALL   DESP                 ;exhibir décimas de hora
0045 58                     POP    AX

0046              HORA3:

0046 04 30                  ADD    AL,'0'               ;convertir a ASCII
0048 E8 0075 R              CALL   DESP                 ;exhibir unidades de horas
004B B0 3A                  MOV    AL,':'               ;exhibir los dos puntos (:)
004D E8 0075 R              CALL   DESP

0050 8A C1                  MOV    AL,CL                ;obtener los minutos
0052 32 E4                  XOR    AH,AH                ;borrar AH
0054 D4 0A                  AAM                         ;convertir a BCD

0056 05 3030                ADD    AX,3030H             ;convertir a ASCII
0059 50                     PUSH   AX
005A 8A C4                  MOV    AL,AH
005C E8 0075 R              CALL   DESP                 ;exhibir décimas de minutos
005F 58                     POP    AX
0060 E8 0075 R              CALL   DESP                 ;exhibir unidades de minutos

0063 B0 20                  MOV    AL,''                ;exhibir espacio
0065 E8 0075 R              CALL   DESP
```

EJEMPLO 6–25 (página 4 de 5)

```
0068 8A C7                   MOV   AL,BH             ;exhibir si es A o P
006A E8 0075 R               CALL  DESP

006D BA 00D2 R               MOV   DX,OFFSET MES1    ;exhibir la M
0070 B4 09                   MOV   AH,9
0072 CD 21                   INT   21H
0074 C3                      RET

0075                   HORA  ENDP
0075                   DESP  PROC  NEAR              ;exhibir ASCII

0075 50                      PUSH  AX
0076 B4 06                   MOV   AH,6
0078 8A D0                   MOV   DL,AL
007A CD 21                   INT   21H
007C 58                      POP   AX
007D C3                      RET

007E                   DESP  ENDP

007E                   FECHA PROC  NEAR              ;exhibir la fecha

007E B4 2A                   MOV   AH,2AH            ;obtener la fecha
0080 CD 21                   INT   21H
0082 52                      PUSH  DX                ;salvar mes y dia
0083 32 E4                   XOR   AH,AH             ;borrar AX
0085 03 C0                   ADD   AX,AX             ;duplicar AX
0087 BE 0000 R               MOV   SI,OFFSET DIA     ;direcccionar tabla del dia
008A 03 F0                   ADD   SI,AX
008C 8B 14                   MOV   DX,[SI]           ;obtener dirección de la cadena
008E B4 09                   MOV   AH,9
0090 CD 21                   INT   21H               ;exhibir el día

0092 5A                      POP   DX
0093 52                      PUSH  DX
0094 8A C6                   MOV   AL,DH             ;obtener el mes
0096 FE C8                   DEC   AL
0098 32 E4                   XOR   AH,AH
009A 03 C0                   ADD   AX,AX
009C BE 000E R               MOV   SI,OFFSET MES     ;direccionar tabla del mes
009F 03 F0                   ADD   SI,AX
00A1 8B 14                   MOV   DX,[SI]           ;obtener dirección de la cadena
00A3 B4 09                   MOV   AH,9
00A5 CD 21                   INT   21H               ;exhibir el mes

00A7 5A                      POP   DX                ;obtener el día
00A8 8A C2                   MOV   AL,DL
00AA 32 E4                   XOR   AH,AH             ;borrar AH
00AC D4 0A                   AAM                     ;convertir a BCD
00AE 0A E4                   OR    AH,AH             ;probar décimas de día
00B0 74 09                   JZ    FECHA1            ;si es cero
00B2 50                      PUSH  AX
00B3 8A C4                   MOV   AL,AH
00B5 04 30                   ADD   AL,'0'            ;convertir a ASCII
00B7 E8 0075 R               CALL  DESP              ;exhibir décimas de día
00BA 58                      POP   AX

00BB                   FECHA1:
00BB 04 30                   ADD   AL,'0'            ;convertir a ASCII
```

EJEMPLO 6–25 (página 5 de 5)

```
00BD E8 0075 R              CALL   DESP              ;exhibir unidades de día
00C0 BA 00D8 R              MOV    DX,OFFSET MES2
00C3 81 F9 07D0             CMP    CX,2000           ;probar año 2000
00C7 72 06                  JB     FECHA2            ;si el año es 19xx
00C9 83 E9 64               SUB    CX,100
00CC BA 00DD R              MOV    DX,OFFSET MES3

00CF               FECHA2:

00CF B4 09                  MOV    AH,9              ;exhibir 19 o 20
00D1 CD 21                  INT    21H
00D3 81 E9 076C             SUB    CX,1900           ;ajustar el año
00D7 8B C1                  MOV    AX,CX
00D9 D4 0A                  AAM                      ;convertir a BCD
00DB 05 3030                ADD    AX,3030H          ;convertir a ASCII
00DE 50                     PUSH   AX
00DF 8A C4                  MOV    AL,AH
00E1 E8 0075 R              CALL   DESP              ;exhibir décimas de año
00E4 58                     POP    AX
00E5 E8 0075 R              CALL   DESP              ;exhibir unidades de año
00E8 C3                     RET

00E9               FECHA    ENDP

00E9               COD      ENDS

                            END    MAIN
```

La hora está disponible en DOS con el empleo de la función 2CH de la INT 21H. Con esto se presentan las horas en CH y los minutos en CL. También están disponibles los segundos en DH y las centésimas de segundo en DL. La fecha está disponible con el empleo de la función 2AH de la INT 21H. Con ello, quedan el día de la semana en AL, el año en CX, el día del mes en DH y el mes en DL.

Con este procedimiento se emplean dos tablas ASCII que convierten el día y al mes en cadenas de caracteres ASCII. También se utiliza la instrucción AAM que convierte binario a BCD para la hora y la fecha. La exhibición de los datos se maneja en dos formas: mediante cadenas de caracteres (función 09H) y con un solo carácter (función 06H).

La memoria consta de tres segmentos: pila (STAC), datos (DAT) y código (COD). El segmento de datos contiene las cadenas de caracteres utilizados con los procedimientos para exhibir hora y fecha. El segmento de datos contiene los procedimientos MAIN, HORA, FECHA y DESP. El procedimiento MAIN es LEJANO porque es el programa o el módulo principal. Los otros procedimientos son CERCANOS o locales porque se utilizan con MAIN.

6–4 ARCHIVOS DE DISCOS

Los datos se encuentran almacenados en el disco en forma de archivos. El disco en sí está organizado en cuatro partes principales: el sector para carga inicial, la tabla de localización de archivos (FAT), el directorio raíz y las zonas para almacenamiento de datos. El primer sector en el disco es el sector de carga inicial, el cual se emplea para cargar el sistema operativo de disco (DOS) desde

el disco hacia la memoria cuando se enciende la computadora. La FAT es donde DOS almacena los nombres de los archivos y subdirectorios y sus localizaciones en el disco. Todas las consultas de cualquier archivo del disco se manejan con la FAT. El directorio de raíces es donde se consultan todos los demás subdirectorios y archivos. Todos los archivos del disco se consideran de acceso secuencial lo cual significa que se accesan un byte a la vez desde el principio hasta el final del archivo.

Organización del disco

En la figura 6-2 se ilustra la organización de los sectores y pistas en la superficie del disco. Esta organización se aplica para disco flexible y disco duro. La pista externa es siempre la pista 0 y la interna es 39 (disquete de doble densidad) o 79 (disquete de alta densidad). La pista interna en un disco duro se determina por el tamaño del disco y podría ser 10000 o mayor para discos duros muy grandes.

En la figura 6-3 se ilustra la organización de los datos en un disco. La longitud de la FAT se determina por el tamaño del disco. Asimismo, la longitud del directorio raíces se determina por el número de archivos y subdirectorios que hay en él. El sector de carga inicial siempre *tiene 512* bytes y está localizado en la pista externa en el sector 0, que es el primero.

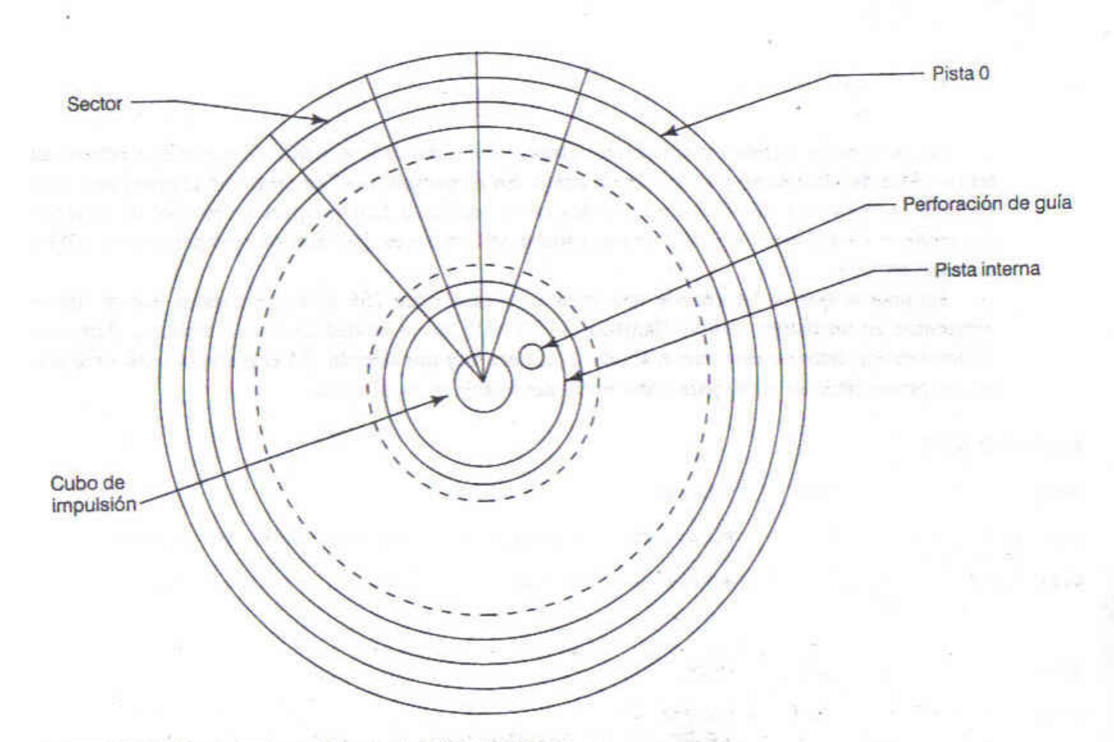


FIGURA 6–2 El formato de un disquete de 5.25 pulgadas.

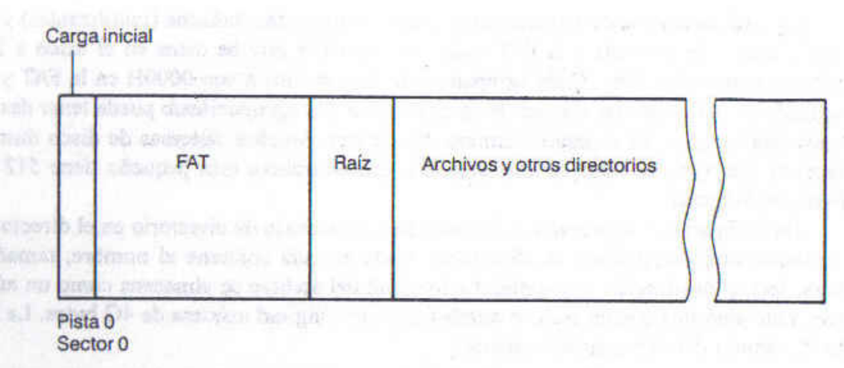


FIGURA 6–3 Las zonas principales de almacenamiento de datos en un disco.

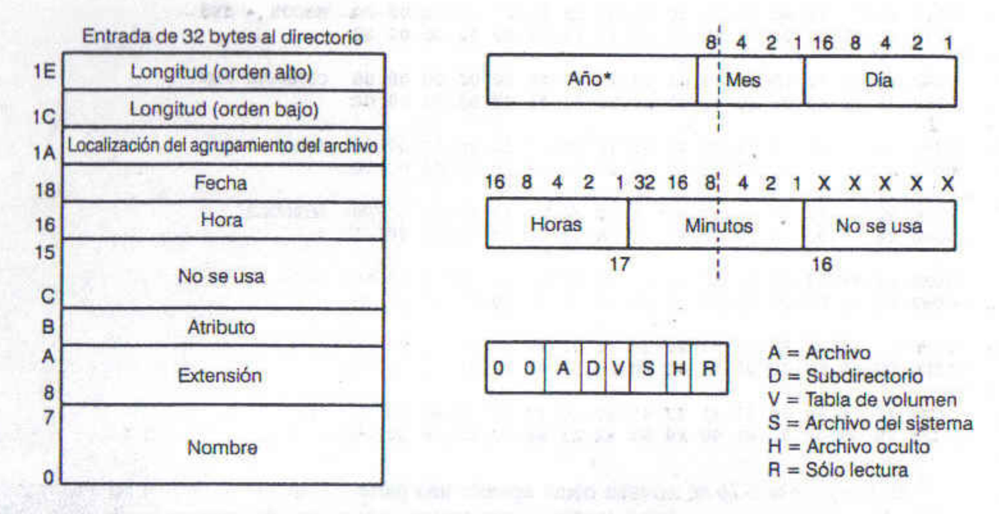


FIGURA 6–4 El formato para cualquier entrada a un directorio o subdirectorio.

El sector de carga inicial contiene un *programa cargador de autoinicialización* que se transfiere a la RAM cuando se enciende el sistema. El cargador inicial automático ejecuta y carga después los programas IO.SYS[3] y MSDOS.SYS[4] en la RAM. Después, este inicializador pasa el control al programa de control del MSDOS y la computadora queda bajo el control del procesador de comandos del DOS.

[3] IO. SYS es un programa de control de E/S que surte Microsoft Corporatiion con el Microsoft DOS.

[4] MSDOS,SYS Es el sistema operativo de discos de Microsoft.

La FAT indica cuáles sectores están libres, cuáles están dañados (inutilizables) y cuáles contienen datos. Se consulta a la FAT cada vez que DOS escribe datos en el disco a fin de poder encontrar un sector libre. Cada agrupamiento libre se indica con 0000H en la FAT y cada sector ocupado se indica por un número de agrupamiento. Un *agrupamiento* puede tener desde un sector hasta una longitud de cualquier número de sectores. Muchos sistemas de disco duro emplean 4 sectores por agrupamiento, lo cual significa que el archivo más pequeño tiene 512 x 4 o 2048 bytes de longitud.

En la figura 6-4 se muestra el formato de cada entrada de directorio en el directorio raíz o en cualquier otro directorio o subdirectorio. Cada entrada contiene el nombre, tamaño, atributo, hora, fecha, localización y longitud. La longitud del archivo se almacena como un número de 32 bits. Esto significa que un archivo puede tener una longitud máxima de 4G bytes. La localización es el número del agrupamiento inicial.

EJEMPLO 6–26

```
0000 49 4F 20 20 20 20 20 20 53 59 53 07 00 00 00 00  IO      SYS
0010 00 00 00 00 00 00 00 00 93 11 02 00 39 82 00 00

0020 4D 53 44 4F 53 20 20 20 53 59 53 07 00 00 00 00  MSDOS   SYS
0030 00 00 00 00 00 00 C0 44 93 12 13 00 92 00 00 00

0040 43 4F 4D 4D 41 4E 44 20 43 4F 4D 00 00 00 00 00  COMMAND COM
0050 00 00 00 00 00 00 00 00 93 11 26 00 B5 92 00 00

0060 42 41 52 52 59 20 42 52 45 59 20 28 00 00 00 00  BARRY BREY
0070 00 00 00 00 00 00 E0 AD 6A 13 00 00 00 00 00 00

0080 50 43 54 4F 4F 4C 53 20 20 20 20 10 00 00 00 00  PCTOOLS
0090 00 00 00 00 00 00 80 AE 6A 13 5C 00 00 00 00 00

00A0 44 4F 53 20 20 20 20 20 20 20 20 10 00 00 00 00  DOS
00B0 00 00 00 00 00 00 E0 B0 6A 13 4E 00 00 00 00 00

00C0 52 55 4E 5F 46 57 20 20 42 41 54 00 00 00 00 00  FUN_FW  BAT
00D0 00 00 00 00 00 00 40 BD 6A 13 97 0F 4A 00 00 00

00E0 46 4F 4E 54 57 41 52 45 20 20 20 10 00 00 00 00  FONTWARE
00F0 00 00 00 00 00 00 60 BD 6A 13 6E 00 00 00 00 00
```

En el ejemplo 6-26 se muestra cómo aparece una parte del directorio de raíz en una expansión (dump) hexadecimal. Trate de determinar la fecha, hora, localización y longitud de cada entrada. Además determine el atributo de cada entrada. En el ejemplo se presentan datos hexadecimales y también datos ASCII, como se acostumbra para la mayor parte de las expansiones de computadoras.

Casi siempre los archivos se accesan por medio de llamadas a funciones de la de DOS INT 21H. Hay dos formas de accesar un archivo con el empleo de la INT 21H. La primera es por medio de un bloque de control de archivo y en la segunda se emplea un manejador de archivos. En la actualidad, todos los programas se accesan por medio de un manejador de archivos, por lo cual en este libro se describe el empleo de manejadores de archivo para tener acceso a ellos. Los bloques de control de archivo son una continuación y modernización de uno de los primeros sistemas operativos llamado CPM (programa de control de micro que se utilizó con los sistemas de 8 bits para computadora basadas en los microprocesadores Z80 u 8080.

Acceso secuencial a archivos

Todos los archivos de DOS son secuenciales. Un archivo secuencial se almacena y accesa desde el comienzo del archivo hasta el final: el primer byte y todos los bytes entre aquél y el último, se deben accesar para leer el último byte. Por fortuna, los archivos se leen y escriben mediante llamadas a la función INT 21H del DOS (véase apéndice A) lo cual facilita el accceso y la "manipulación". En esta sección se describe la forma de abrir, leer, escribir, borrar y cambiar el nombre de un archivo de acceso secuencial.

Creación de un archivo. Antes de poder leer un archivo, éste ya debe estar en el disco. Para crear un archivo se emplea la 3CH función 3CH de la INT 21H. El nombre del archivo se debe almacenar en una localidad direccionada por DS:DX antes de llamar a la función y CX debe contener el atributo del archivo (o subdirectorio) creado.

EJEMPLO 6-27

```
0000                    DAT     SEGMENT

0000 44 4F 47 2E 54 58          FILE1  DB   'DOG.TXT',0            ;nombre de archivo DOG.TXT
     54 00
0008 43 3A 44 41 54 41          FILE2  DB   'C:DATA.DOC',0         ;archivo c:data.doc
     2E 44 4F 43 00
0013 43 3A 5C 44 52 45          FILE3  DB   'C:\DREAD\ERROR.FIL',0 ;archivo C:\DREAD\ERROR.FIL
     41 44 5C 45 52 52
     4F 52 2E 46 49 4C
     00

0026                    DAT     ENDS
```

Un *nombre de archivo* se almacena siempre como una cadena ASCII-Z y puede contener las trayectorias del directorio y la unidad de disco. En el ejemplo 6-27 se muestran algunos nombres de archivo en cadena ASCII-Z almacenados en un segmento de datos para acceso por las utilerías del archivo. Una *cadena ASCII-Z* es una cadena de caracteres que termina siempre con un 00H o un carácter nulo.

Supóngase que se ha llenado una zona de memoria de 256 bytes, con datos que se deben almacenar en un nuevo archivo llamado DATO.NUV en la unidad de disco implícita. Antes de poder escribir datos en este nuevo archivo, primero hay que crearlo. En el ejemplo 6-28 se muestra un procedimiento corto para crear este nuevo archivo en el disco.

EJEMPLO 6–28

```
0000                    DAT     SEGMENT

0000 44 41 54 41 2E 4E          FILE1  DB    'DATA.NEW',0    ;archivo nombre del DATO.UV
     45 57 00
0009 0100[                      BUFFER DB    256 DUP (?)     ;memoria de datos buffer
              ??
                 ]

0109                    DAT     ENDS

0000                    COD     SEGMENT
                                ASSUME CS:COD,DS:DAT
```

```
0000 B8 ---- R              MOV    AX,DAT
0003 8E D8                  MOV    DX,AX              ;cargar DS

0005 B4 3C                  MOV    AH,3CH             ;cargar función crear
0007 33 C9                  XOR    CX,CX              ;00H = atributo
0009 BA 0000 R              MOV    DX,OFFSET FILE1    ;direccionar nombre de ASCII-Z
000C CD 21                  INT    21H                ;abrir DATO.UV

000E 72 20                  JC     ERROR              ;error al abrir
                            .      .
                            .      .
                            .      .
0030              COD       ENDS

                            END
```

Siempre que se crea un archivo, el registro CX debe contener los atributos o características del archivo. En la tabla 6-3 se enumeran las posiciones de bits de los atributos y se los define. Un 1 lógico en un bit selecciona el atributo; un 0 lógico no lo hace.

Después de regresar de la INT 21H, la bandera de acarreo indica si ocurrió un error (CF = 1) durante la apertura del archivo. Pueden ocurrir algunos errores que se localizan, si se necesita, con la llamada número 59H de función INT 21H y que son: no se encontró la trayectoria, no hay manejadores de archivo disponibles, o hay un error en el medio de almacenamiento. Si la bandera de acarreo es cero, no ha ocurrido error y el registro AX contiene un manejador de archivo, el cual es un número que se utiliza para consultar el archivo después de abrirlo. El **manejador de archivo** permite accesar a un archivo sin emplear el nombre de cadena ASCII-Z del archivo, lo cual acelera el trabajo.

Escritura en un archivo. Una vez que hemos creado el nuevo archivo el llamado ARCHI.UV, ya se pueden escribir datos en él. Lo primero que se necesita para escribir en un archivo, es abrirlo. Cuando se abre un archivo, el manejador de archivo retorna en el registro AX. El manejador de archivo se utiliza para consultar el archivo siempre que se escriben datos.

La función 40H se utiliza para escribir datos en un archivo recién abierto. Además de cargar un 40H en AH, hay que cargar BX = el manejador de archivo, CX = el número de bytes que se van a escribir y DS:DX= la dirección del área zona en que se va a escribir en el disco.

TABLA 6-3 Definiciones de atributos de archivo

Posición de bit	*Atributo*	*Función*
0	Sólo lectura	Archivo o subdirectorio de sólo lectura
1	Oculto	Evita que el nombre del archivo o subdirectorio aparezca en el directorio cuando se da DIR desde la línea de comandos del DOS
2	Sistema	Especifica a un archivo como archivo del sistema
3	Volumen	Especifica la etiqueta de volumen del disco
4	Subdirectorio	Especifica un nombre de subdirectorio
5	Archivo muerto	Indica que se ha cambiado un archivo y que se debe pasar a archivo muerto

EJEMPLO 6–29

```
                              .      .
                              .      .
                              .      .
0010 8B D8                    MOV    BX,AX              ;mover manejador a BX
0012 B4 40                    MOV    AH,40H             ;cargar función de escritura
0014 B9 0100                  MOV    CX,256             ;cargar conteo
0017 BA 0009 R                MOV    DX,OFFSET BUFFER   ;direccionar BUFFER
001A CD 21                    INT    21H                ;escribir 256 bytes de BUFFER

001C 72 32                    JC     ERROR1             ;error en la escritura
```

Supóngase que se necesita escribir 256 bytes de la memoria BUFFER al archivo. Para lograrlo, se emplea la función 40H como se ilustra en el ejemplo 6-29. Si ocurre un error durante una operación de escritura, se activa la bandera de acarreo. Si no ocurre error, se desactiva la bandera de acarreo y el número de bytes escritos en el archivo se retorna en el registro AX. Los errores que ocurren en la escritura suelen indicar que el disco está lleno o que hay algún tipo de error en el medio de almacenamiento.

Apertura, lectura y cierre de un archivo. Lo primero que se necesita para leer un archivo, es abrirlo. Cuando se abre un archivo, DOS examina el directorio para determinar si ese archivo existe y retorna el manejador de archivo del DOS al registro AX. El manejador de archivo del DOS se debe emplear para leer, escribir y cerrar un archivo.

EJEMPLO 6–30

```
0000                   LEER   PROC   NEAR

0000 B4 3D                    MOV    AH,3DH             ;cargar función de apertura
0002 B0 00                    MOV    AL,0               ;seleccionar lectura
0004 BA 0000 R                MOV    DX,OFFSET ARCH1    ;direccionar nombre del archivo
0007 CD 21                    INT    21H                ;abrir archivo

0009 72 15                    JC     ERROR              ;hay error

000B 8B D8                    MOV    BX,AX              ;transferir manejador de archivo a BX

000D B4 3F                    MOV    AH,3FH             ;cargar función de lectura
000F B9 0100                  MOV    CX,256             ;número de bytes
0012 BA 0009 R                MOV    DX,OFFSET BUFFER   ;direccionar BUFFER
0015 CD 21                    INT    21H                ;leer 256 bytes

0017 72 07                    JC     ERROR              ;hay error
0019 B4 3E                    MOV    AH,3EH             ;cargar función de cerrar
001B CD 21                    INT    21H                ;cerrar el archivo

001D 72 01                    JC     ERROR              ;hay error

001F C3                       RET

0020                   LEER   ENDP
```

En el ejemplo 6-30 se muestra una secuencia de instrucciones que abren un archivo leen y cargan 256 bytes de la zona de memoria BUFFER y cierran el archivo. Cuando se abre un archivo (AH = 3DH), el registro AL especifica el tipo de operación permitida para ese archivo abierto. Si AL = 00H, se abre el archivo para lectura; si AL = 01H, se abre el archivo para escritura; si AL = 02H, se abre el archivo para lectura o escritura.

La función 3FH sirve para leer un archivo. Igual que con la función de escritura, BX contiene el manejador de archivo , CX el número de bytes que se van a leer y DS:DX contiene la ubicación de una zona de memoria en donde están almacenados los datos. Igual que con todas las funciones de discos, la bandera de acarreo indica un error cuando es un 1 lógico; si es un 0 lógico, el registro AX indica el número de bytes que se leyeron en el archivo.

Cerrar el archivo es muy importante. Si se deja abierto un archivo, pueden ocurrir problemas serios que pueden destruir el disco y todos los datos que contiene. Si se escribe un archivo, pero no se cierra, la FAT puede "corromperse" y hará difícil o imposible recuperar datos del disco. Después de leer o escribir un disco, no olvide cerrarlo.

El apuntador de archivo. Cuando se abre, escribe o lee un archivo, el apuntador de archivo direcciona la localidad actual en el archivo secuencial. Cuando se abre un archivo, el apuntador de archivo siempre direcciona el primer byte de un archivo. Si el archivo tiene una longitud de 1024 bytes y una función de lectura lee 1023 bytes, el apuntador de archivo direcciona al último byte del archivo pero no el final del archivo.

El **apuntador de archivo** es un número de 32 bits que direcciona a cualquier byte de un archivo. Una vez abierto un archivo, el apuntador se puede mover con la función 42H para desplazar el apuntador de archivo. Un apuntador de archivo se puede mover desde el comienzo del archivo (AL = 00H) desde la ubicación en curso (AL = 01H) o desde el final del archivo (AL = 02H). En la práctica, se utilizan los tres sentidos del movimiento para accesar a diferentes partes del archivo. La distancia en que se mueve el apuntador de archivo la especifican los registros CX y DX. El registro DX contiene la parte menos significativa y el CX la parte más significativa de la distancia. El registro BX debe contener el manejador de archivo antes de emplear la función 42H para mover el apuntador

Supóngase que hay un archivo en el disco y que se deben agregar 256 bytes de nueva información. Cuando se abre el archivo, el apuntador de archivo direcciona al primer byte del archivo. Si se intenta escribir sin mover el apuntador hasta el final del archivo, los nuevos datos quedarán sobrescritos en los primeros 256 bytes del archivo. En el ejemplo 6-31 se muestra un procedimiento que abre un archivo, mueve el apuntador hasta el final del archivo, escribe 256 bytes de datos y, luego, cierra el archivo. Con esto, *se agregan* al archivo 256 bytes de datos nuevos.

EJEMPLO 6–31

```
0000                    AGREGA  PROC   NEAR

0000 B4 3D                      MOV    AH,3DH               ;cargar función de apertura
0002 B0 01                      MOV    AL,1                 ;seleccionar escritura
0004 BA 0000 R                  MOV    DX,OFFSET FILE1      ;direccionar nombre del archivo
0007 CD 21                      INT    21H                  ;abrir archivo

0009 72 21                      JC     ERROR                ;hay error

000B 8B D8                      MOV    BX,AX                ;mover manejador de archivo a BX
```

```
000D B4 42                 MOV   AH,42H             ;cargar función mover apuntador de archivo
000F B0 02                 MOV   AL,02H             ;mover desde el final
0011 33 C9                 XOR   CX,CX              ;mover a 0 bytes desde el final
0013 33 D2                 XOR   DX,DX
0015 CD 21                 INT   21H                ;mover apuntador hasta el final

0017 72 13                 JC    ERROR              ;hay error

0019 B4 40                 MOV   AH,40H             ;cargar función de escritura
001B B9 0100               MOV   CX,256             ;número de bytes
001E BA 0009 R             MOV   DX,OFFSET BUFFER   ;direccionar BUFFER
0021 CD 21                 INT   21H                ;escribir 256 bytes

0023 72 07                 JC    ERROR              ;hay error

0025 B4 3E                 MOV   AH,3EH             ;cargar función de cerrar
0027 CD 21                 INT   21H                ;cerrar archivo

0029 72 01                 JC    ERROR              ;hay error

002B C3                    RET

002C               AGREGA  ENDP
```

Una de las maniobras más difíciles en un archivo es introducir nuevos datos en el centro del archivo. En la figura 6-5 se ilustra cómo se logra con la apertura de un segundo archivo. Se debe tener en cuenta que la parte del archivo antes del punto de introducción, se copia en el nuevo archivo. Esto va seguido por la nueva información antes de que se añada el resto del archivo después de la introducción en el nuevo archivo. Una vez que el archivo nuevo está completo, se borra el archivo viejo y al archivo nuevo se le cambia el nombre por el del archivo viejo.

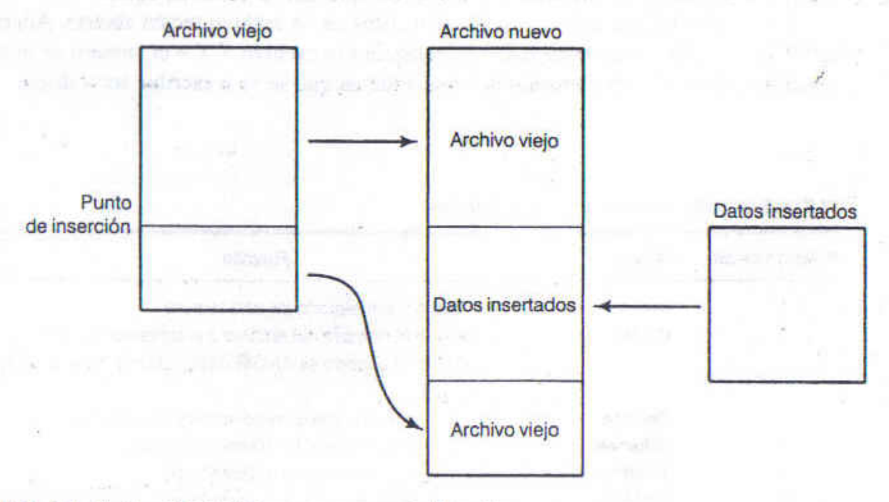


FIGURA 6-5 Introducción de datos nuevos en un archivo viejo.

En el ejemplo 6-32 se muestra un procedimiento para introducir nuevos datos, hasta de 64K bytes de longitud, en un archivo viejo. Este procedimiento requirió algunos parámetros de entrada para que fuera de uso general. El primer parámetro es el nombre del archivo viejo que se pasa al procedimiento por medio del registro DS:DX. El segundo parámetro (registro CX) es el número de nuevos bytes de información que se escribirán en el archivo nuevo. El tercer parámetro es la ubicación (ES:SI) de los nuevos datos que se introducirán en el archivo. El último parámetro fue el punto de introducción ubicado en los registros BP y DI, en donde BP contiene el punto de introducción más significativo y DS el menos.

EJEMPLO 6–32 (página 1 de 3)

```
0000                    DATOS    SEGMENT PUBLIC

0000 0100[                       BUFFER  DB   256 DUP (?)   ;buffer
          ??
             ]
0100 0000                        TEMP    DW   ?             ;datos temporales
0102 0000                        TEMP1   DW   ?             ;datos temporales
0104 0000                        MAN_VIE DW   ?             ;manejador viejo
0106 0000                        MAN_NUV DW   ?             ;manejador temporal
0108 54 45 4D 50 2E 24           TEMPS   DB   'TEMP.$$$',0  ;archivo temporal
     24 24 00

0111                    DATOS    ENDS

0000                    CODE     SEGMENT

                                 ASUMME CS:CODE,DS:DATA

0000                    INSERTA  PROC    FAR

0000 89 16 0100 R                MOV     TEMP,DX            ;salvar dirección de archivo viejo
0004 89 0E 0102 R                MOV     TEMP1,CX           ;salvar el conteo

0008 B8 3D02                     MOV     AX,3D02H           ;abrir archivo viejo
000B CD 21                       INT     21H
000D A3 0104 R                   MOV     MAN_VIE,AX         ;salvar manejador de archivo viejo
0010 73 05                       JNC     INSERTA1           ;si no hay error

0012 E8 009F R                   CALL    C_VIE              ;cerrar archivo viejo
0015 F9                          STC                        ;indicar error
0016 CB                          RET

0017                    INSERTA1:

0017 B4 3C                       MOV     AH,3CH             ;abrir archivo temp
0019 33 C9                       XOR     CX,CX
001B BA 0108 R                   MOV     DX,OFFSET TEMPS
001E CD 21                       INT     21H
0020 A3 0106 R                   MOV     MAN_NUV,AX         ;salvar manejador de archivo temp
0023 73 08                       JNC     INSERTA2

0025                    SALIDA:

0025 E8 00A8 R                   CALL    C_TEMP             ;cerrar archivo temp
0028 E8 009F R                   CALL    C_VIE              ;cerrar archivo viejo
002B F9                          STC                        ;indicar error
```

EJEMPLO 6–32 (página 2 de 3)

```
002C CB                    RET
002D              INSERTA2:
002D 81 FF 0100            CMP   DI,256         ;probar punto de introducción
0031 77 07                 JA    INSERTA3       ;si es mayor de 256
0033 0B ED                 OR    BP,BP
0035 75 03                 JNE   INSERTA3       ;si es mayor de 256
0037 EB 17 90              JMP   INSERTA4       ;si es menor de 256

003A              INSERTA3:

003A B9 0100               MOV   CX,256
003D E8 00B1 R             CALL  R_OLD          ;leer archivo viejo
0040 72 E3                 JC    EXIT           ;hay error
0042 E8 00BD R             CALL  W_TEMP         ;escribir archivo temporal
0045 72 DE                 JC    EXIT           ;hay error

0047 81 EF 0100            SUB   DI,256         ;decrementar BP-DI en 256
004B 83 DD 00              SBB   BP,0
004E EB DD                 JMP   INSERTA2

0050              INSERT4:

0050 8B CF                 MOV   CX,DI          ;obtener conteo
0052 E8 00B1 R             CALL  R_VIE          ;leer archivo viejo
0055 72 CE                 JC    EXIT           ;hay error
0057 E8 00BD R             CALL  W_TEMP         ;escribir archivo temporal
005A 72 C9                 JC    EXIT           ;hay error

005C 8B 0E 0102 R          MOV   CX,TEMP1       ;obtener conteo de introducción
0060 1E                    PUSH  DS             ;salvar segmento de datos
0061 8C C0                 MOV   AX,ES
0063 8E D8                 MOV   DS,AX          ;cargar DS con ES

0065 8B D6                 MOV   DX,SI          ;obtener dirección
0067 B4 40                 MOV   AH,40H         ;escribir datos de introducción
0069 CD 21                 INT   21H
006B 1F                    POP   DS             ;recuperar DS
006C 72 B7                 JC    SALIDA         ;hay error

006E              INSERT5:

006E B9 0100               MOV   CX,256         ;escribir el resto del archivo
0071 E8 00B1 R             CALL  R_VIE          ;leer archivo viejo
0074 72 AF                 JC    SALIDA         ;hay error
0076 0B C0                 OR    AX,AX          ;probar si es final de archivo
0078 74 05                 JE    INSERTA6       ;si es final de archivo
007A E8 00BD R             CALL  W_TEMP         ;escribir archivo temporal
007D EB EF                 JMP   INSERTA5       ;repetir hasta llegar al final

007F              INSERT6:

007F E8 009F R             CALL  C_VIE          ;cerrar archivo viejo
0082 E8 00A8 R             CALL  C_TEMP         ;cerrar archivo temporal
0085 8B 16 0100 R          MOV   DX,TEMP        ;borrar archivo viejo
0089 B4 41                 MOV   AH,41H
008B CD 21                 INT   21H

008D 06                    PUSH  ES             ;salvar ES
008E 8C D8                 MOV   AX,DS
```

EJEMPLO 6–32 (página 3 de 3)

```
0090 8E C0                    MOV    ES,AX              ;cargar ES con DS

0092 8B 3E 0100 R             MOV    DI,TEMP            ;obtener nombre de archivo viejo
0096 BA 0108 R                MOV    DX,OFFSET TEMPS    ;cambiar nombre al archivo
0099 B4 56                    MOV    AH,56H
009B CD 21                    INT    21H
009D 07                       POP    ES
009E CB                       RET

009F                 INSERTA  ENDP

009F                 C_VIE    PROC   NEAR
                                                        ;cerrar archivo viejo
009F 8B 1E 0104 R             MOV    BX,OLD_HAN
00A3 B4 3E                    MOV    AH,3EH
00A5 CD 21                    INT    21H
00A7 C3                       RET

00A8                 C_VIE    ENDP
                                                        ;cerrar archivo temporal
00A8                 C_TEMP   PROC   NEAR

00A8 8B 1E 0106 R             MOV    BX,NEW_HAN
00AC B4 3E                    MOV    AH,3EH
00AE CD 21                    INT    21H
00B0 C3                       RET

00B1                 C_TEMP   ENDP

00B1                 L_VIE    PROC   NEAR

00B1 8B 1E 0104 R             MOV    BX,OLD_HAN
00B5 BA 0000 R                MOV    DX,OFFSET BUFFER
00B8 B4 3F                    MOV    AH,3FH
00BA CD 21                    INT    21H
00BC C3                       RET

00BD                 L_VIE    ENDP
00BD                 E_TEMP   PROC   NEAR

00BD 8B C8                    MOV    CX,AX
00BF 8B 1E 0106 R             MOV    BX,MAN_NUV
00C3 BA 0000 R                MOV    DX,OFFSET BUFFER
00C6 B4 40                    MOV    AH,40H
00C8 CD 21                    INT    21H
00CA C3                       RET

00CB                 E_TEMP   ENDP

00CB                 CODE     ENDS

                              END
```

En este procedimiento se utilizan dos nuevas funciones de la INT 21H. Las funciones de borrar y cambiar el nombre se emplean para borrar el archivo viejo antes de que al archivo temporal se le cambie el nombre y se le ponga el del viejo.

Archivos de acceso aleatorio

Los archivos de acceso aleatorio se producen con programas que emplean archivos de acceso secuencial. El archivo de acceso aleatorio se direcciona con un número de grabación en vez de tener que buscar todos los datos en el archivo. La función de mover el apuntador se vuelve muy importante cuando se abren archivos de acceso aleatorio; éstos son mucho más fáciles de usar con grandes volúmenes de datos.

Cómo crear un archivo de accceso aleatorio. La planeación adelantada es de máxima importancia para abrir o producir un sistema de archivo de acceso aleatorio. Supóngase que se requiere un archivo de acceso aleatorio para archivar los nombres de clientes. Cada grabación para un cliente requiere 16 bytes para el apellido, 16 bytes para el nombre y 1 byte para la inicial intermedia. Además, la grabación para cada cliente tiene dos líneas de 32 bytes cada una para el nombre de la calle, una línea de 16 bytes para la ciudad o población, 2 bytes para la abreviatura (en su caso) del nombre del estado o provincia y 9 bytes para el código postal. O sea, que sólo para la información básica del cliente se necesitan 105 bytes. La información adicional amplía la grabación a una longitud de 256 bytes. Como el negocio está creciendo, se debe prever un archivo para 5,000 clientes. Esto significa que la longitud total del archivo de acceso aleatorio es de 1 280 000 bytes.

EJEMPLO 6–33

```
0000                    DATOS   SEGMENT

0000 0100[                      BUFFER  DB   256 DUP (0)      ;memoria buffer de 00H
            00
              ]
0100 41 3A 43 55 53 54          ARCH    DB   'A:CLIENT.FIL',0 ;nombre del archivo
     2E 46 49 4C 00

010B                    DATOS   ENDS

0000                    CODE    SEGMENT

                                ASSUME CS:CODE,DS:DATA

0000                    TAREA   PROC    FAR

0000 B8 ---- R                  MOV     AX,DATA          ;cargar DS
0003 8E D8                      MOV     DS,AX

0005 B4 3C                      MOV     AH,3CH           ;producir CLNT.FIL
0007 33 C9                      XOR     CX,CX
0009 BA 0100 R                  MOV     DX,OFFSET ARCHI
000C CD 21                      INT     21H

000E 8B D8                      MOV     BX,AX            ;manejador a BX

0010 BF 1388                    MOV     DI,5000          ;cargar conteo de grabaciones

0013                    TAREA1:

0013 B4 40                      MOV     AH,40H           ;escribir una grabación de 00H
0015 B9 0100                    MOV     CX,256
0018 BA 0000 R                  MOV     DX,OFFSET BUFFER
001B CD 21                      INT     21H
```

```
001D 4F                 DEC   DI
001E 75 F3              JNZ   TAREA1          ;repetir 5000 veces

0020 B4 3E              MOV   AH,3EH          ;cerrar archivo
0022 CD 21              INT   21H

0024 B4 4C              MOV   AH,4CH          ;salir a DOS
0026 CD 21              INT   21H

0028            MAKE    ENDP

0028            CODE    ENDS

                        END   MAKE
```

En el ejemplo 6-33 se ilustra un programa corto para producir un archivo llamado CLIENT.FIL e inserta 5,000 grabaciones en blanco de 256 bytes cada una. Un registro o grabación en blanco contiene 00H en cada byte. Parece ser un archivo muy grande, pero cabe en un solo disquete de alta densidad, de 3.5 o de 5.25 pulgadas; en la práctica en este programa se supone que el disco está en la unidad de disco A.

Lectura y escritura de una grabación. Siempre que se debe leer una grabación, su número se carga en el registro BP y se llama al procedimiento que aparece en el ejemplo 6-34. En este procedimiento se supone que ARCHIVO contiene el número del manejador y que el archivo CLIENT.FIL permanece abierto en todo momento.

EJEMPLO 6–34

```
0000                LEER    PROC  FAR

0000 8B 1E 0100 R           MOV   BX,ARCH           ;obtener manejador
0004 B8 0100                MOV   AX,256            ;multiplicar por 256
0007 F7 E5                  MUL   BP
0009 8B CA                  MOV   CX,DX
000B 8B D0                  MOV   DX,AX
000D B8 4200                MOV   AX,4200H          ;mover apuntador
0010 CD 21                  INT   21H

0012 B4 3F                  MOV   AH,3FH            ;leer la grabación
0014 B9 0100                MOV   CX,256
0017 BA 0000 R              MOV   DX,OFFSET BUFFER
001A CD 21                  INT   21H
001C CB                     RET

001D                LEER    ENDP
```

Se verá que el número de grabación se multiplica por 256 para obtener un conteo para la función de movimiento del apuntador. En cada caso, el apuntador se mueve desde el inicio del archivo hasta la grabación deseada antes de poder leerlo en el área BUFFER de la memoria. Aunque no se ilustra, la escritura se efectúa en la misma forma que la lectura.

6–5 EJEMPLOS DE PROGRAMAS

Una vez descritos los bloques de construcción básicos de la programación, se presentan algunos ejemplos de programas de aplicación. Aunque parezcan carecer de importancia, ofrecen algunas técnicas adicionales para programación e ilustran estilos de programación para el microprocesador 80286.

Programa para calcular

Este programa demuestra cómo la conversión de datos tiene parte importante en muchos programas de aplicación. En el ejemplo 6-35 se ilustra un programa que acepta dos números y los suma, resta, multiplica o divide. Para reducir la complejidad del programa, los números están limitados a cifras de 2 dígitos. Por ejemplo, si se teclea 12 + 24 =, el programa calculará el resultado y exhibirá el 36 como respuesta. Para simplificar todavía más el programa, a los números 0 a 9 se les debe dar entrada como números de dos dígitos: 00 a 09.

EJEMPLO 6–35 (página 1 de 4)

```
0000                    STAC    SEGMENT STACK

0000 0400[                      DW      1024 DUP (?)         ;inicializar la pila
            ????
                        ]

0800                    STAC    ENDS

0000                    DATOS   SEGMENT

0000 0D 0A 24           MES1    DB      13,10,'$'

0003                    DATOS   ENDS

0000                    CODE    SEGMENT

                                ASSUME CS:CODE,DS:DATA,SS:STAC

0000                    PRAL    PROC    FAR
0000 B8 ---- R                  MOV     AX,DATA              ;cargar DS
0003 8E D8                      MOV     DS,AX

0005                    PRAL1:

0005 E8 0069 R                  CALL    NUEVO                ;obtener nueva línea

0008 E8 0071 R                  CALL    LEE                  ;obtener primer número

000B                    PRAL2:

000B E8 0097 R                  CALL    TECLA                ;obtener la operación
000E 3C 2D                      CMP     AL,'-'
0010 74 0C                      JE      PRAL3                ;operación correcta
0012 3C 2B                      CMP     AL,'+'
0014 74 08                      JE      PRAL3                ;operación correcta
```

EJEMPLO 6–35 (página 2 de 4)

```
0016 3C 2F                     CMP    AL,'/'
0018 74 04                     JE     PRAL3             ;operación correcta
001A 3C 2A                     CMP    AL,'*'
001C 75 ED                     JNE    PRAL2             ;operación inválida

001E               PRAL3:

001E 8A D0                     MOV    DL,AL             ;operación con eco
0020 CD 21                     INT    21H

0022 8A D8                     MOV    BL,AL             ;salvar la operación
0024 8B F5                     MOV    SI,BP             ;salvar el primer número
0026 E8 00A0 R                 CALL   SPACE
0029 E8 0071 R                 CALL   READ              ;obtener segundo número

002C               MAIN4:

002C E8 0097 R                 CALL   KEY               ;obtener el igual
002F 3C 3D                     CMP    AL,'='
0031 75 F9                     JNE    MAIN4             ;si no es igual

0033 8A D0                     MOV    DL,AL             ;eco es igual
0035 CD 21                     INT    21H
0037 E8 00A0 R                 CALL   SPACE

003A 80 FB 2B                  CMP    BL,'+'            ;probar para +
003D 74 11                     JE     PLUS
003F 80 FB 2D                  CMP    BL,'-'            ;probar para -
0042 74 13                     JE     MIN
0044 80 FB 2F                  CMP    BL,'/'            ;probar para /
0047 74 15                     JE     DIVS

0049 8B C5                     MOV    AX,BP             ;multiplicar
004B F7 E6                     MUL    SI
004D EB 15 90                  JMP    MAIN5             ;exhibir el producto

0050               PLUS:

0050 8B C5                     MOV    AX,BP             ;sumar
0052 03 C6                     ADD    AX,SI
0054 EB 0E 90                  JMP    MAIN5             ;exhibir la suma

0057               MIN:

0057 8B C6                     MOV    AX,SI             ;restar
0059 2B C5                     SUB    AX,BP
005B EB 07 90                  JMP    MAIN5             ;exhibir la diferencia

005E               DIVS:

005E 33 D2                     XOR    DX,DX             ;dividir
0060 8B C6                     MOV    AX,SI
0062 F7 F5                     DIV    BP                ;exhibir el cociente

0064               MAIN5:

0064 E8 00A7 R                 CALL   DISP              ;exhibir el resultado
0067 EB 9C                     JMP    MAIN1             ;repetir siempre
```

EJEMPLO 6–35 (página 3 de 4)

```
0071 33 ED                  XOR    BP,BP          ;borrar el número
0073 B9 0002                MOV    CX,2           ;cargar conteo
0076 B7 0A                  MOV    BH,10          ;cargar 10

0078               LEE1:

0078 E8 0097 R              CALL   KEY            ;leer tecla
007B 3C 30                  CMP    AL,'0'         ;probar si es número
007D 72 F9                  JB     READ1          ;si no es número
007F 3C 39                  CMP    AL,'9'
0081 77 F5                  JA     READ1          ;si no es número

0083 8A D0                  MOV    DL,AL          ;eco a la pantalla
0085 CD 21                  INT    21H

0087 32 E4                  XOR    AH,AH
0089 2C 30                  SUB    AL,'0'

008B 95                     XCHG   BP,AX          ;formar el número
008C F6 E7                  MUL    BH
008E 95                     XCHG   BP,AX
008F 03 E8                  ADD    BP,AX

0091 E2 E5                  LOOP   LEE1           ;repetir dos veces

0093 E8 00A0 R              CALL   ESPACIO        ;exhibir espacio
0096 C3                     RET

0097               LEE      ENDP

0097               TECLA    PROC   NEAR

0097 B4 06                  MOV    AH,6           ;leer tecla
0099 B2 FF                  MOV    DL,0FFH
009B CD 21                  INT    21H
009D 74 F8                  JE     KEY
009F C3                     RET

00A0               TECLA    ENDP
00A0               ESPACIO  PROC   NEAR

00A0 B4 06                  MOV    AH,6           ;exhibir espacio
00A2 B2 20                  MOV    DL,''
00A4 CD 21                  INT    21H
00A6 C3                     RET

00A7               ESPACIO  ENDP
0069               MAIN     ENDP
0069               NEW      PROC   NEAR

0069 B4 09                  MOV    AH,9           ;exhibir línea nueva
006B BA 0000 R              MOV    DX,OFFSET MES1
006E CD 21                  INT    21H
0070 C3                     RET

0071               NEW      ENDP

0071               READ     PROC   NEAR
```

EJEMPLO 6–35 (página 4 de 4)

```
00A7                    DESP    PROC    NEAR

00A7 33 C9                      XOR     CX,CX           ;borrar conteo
00A9 BB 000A                    MOV     BX,10           ;cargar 10
00AC 3D 8000                    CMP     AX,8000H        ;probar si es negativo
00AF 72 0A                      JB      DISP1           ;si es positivo
00B1 F7 D8                      NEG     AX              ;hacerlo positivo
00B3 50                         PUSH    AX
00B4 B2 2D                      MOV     DL,'-'          ;exhibir -
00B6 B4 06                      MOV     AH,6
00B8 CD 21                      INT     21H
OOBA 58                         POP     AX

00BB                    DESP1:

00BB 33 D2                      XOR     DX,DX
00BD F7 F3                      DIV     BX              ;dividir entre 10
00BF 52                         PUSH    DX              ;salvar el residuo
00C0 41                         INC     CX              ;contar los dígitos
00C1 0B C0                      OR      AX,AX           ;probar el cociente
00C3 75 F6                      JNE     DESP1           ;repetir hasta que sea 0

00C5 B4 06                      MOV     AH,6

00C7                    DESP2:

00C7 5A                         POP     DX              ;exhibir digito
00C8 80 C2 30                   ADD     DL,'O'
00CB CD 21                      INT     21H
00CD E2 F8                      LOOP    DESP2           ;repetir
00CF C3                         RET

00D0                    DESP    ENDP

00D0                    CODE    ENDS

                                END     PRAL
```

Se debe tener en cuenta que este programa es un lazo infinito. Cuando lo haya probado y encontrado que sí funciona, se oprimen las teclas Control (Ctrl) y Alterno (Alt) y luego se oprime la tecla suprimir (Del) para limpiar y volver a cargar la computadora y el DOS. Si lo desea, puede poner una función de salida en el programa. Este programa no soporta la tecla de retroceso para corregir un error al teclear. No se ha intentado tener recuperación después de los errores.

Programa para clasificar números

En ocasiones, hay que clasificar los números por orden. A menudo se logra con una clasificación de burbuja. En la figura 6-6 se presentan cinco números que se clasificaron con una clasificación de burbuja. Se verá que este grupo de 5 números se prueba 4 veces con 4 pasadas. En cada pasada, se comparan y, a veces, intercambian, 2 números consecutivos. Además, se debe tener en cuenta que durante la primera pasada hay 4 comparaciones; durante la segunda hay 3, etcétera.

FIGURA 6–6 Una clasificación de burbuja mostrando los datos conforme se clasifican. Nota: La clasificación de cinco números puede requerir cuatro pasadas.

```
6 ◄┐6      6      6      6
9 ◄┘9 ◄┐   1      1      1
1      1 ◄┘9 ◄┐   3      3      Pasada 1
3      3      3 ◄┘9 ◄┐   2
2      2      2      2 ◄┘9

6 ◄┐1      1      1
1 ◄┘6 ◄┐   3      3
3      3 ◄┘6 ◄┐   2             Pasada 2
2      2      2 ◄┘6
9      9      9      9

1 ◄┐1      1
3 ◄┘3 ◄┐   2
2      2 ◄┘3                    Pasada 3
6      6      6
9      9      9
```

En el ejemplo 6-36 se ilustra un programa que acepta 10 números (0 a 65535) desde el teclado. Después que se aceptan estos números de 16 bits y se almacenan en la sección ARREGLO de la memoria, se clasifican mediante la técnica de clasificación por burbuja, en la cual se utiliza una bandera para determinar si todos los números se intercambiaron en una pasada. Si no se intercambió ningún número, es que está en orden y termina la clasificación.

EJEMPLO 6–36 (página 1 de 4)

```
0000                          STAC    SEGMENT STACK

0000 0400[                            DW      1024 DUP (?)        ;inicializar la pila
               ????
                    ]

0800                          STAC    ENDS

0000                          DATOS   SEGMENT

0000 000A[                            ARRAY   DW   10 DUP (?)     ;espacio para números
               ????
                    ]
0014 0D 0A 24                         MES1    DB   13,10,'$'
0017 0D 0A 45 6E 74 65                MES2    DB   13,10,'Dar entrada a 10 números:'13,100,'$'
     72 20 31 30 20 6E
     75 6D 62 65 72 73
     3A 0D 0A 24
002D 0D 0A 53 6F 72 74                MES3    DB   13,10,'Datos clasificados:',13,10,'$'
     65 64 20 44 61 74
     61 3A 0D 0A 24

003E                          DATOS   ENDS

0000                          CODE    SEGMENT

                                      ASSUME CS:CODE,DS:DATA,SS:STAC

0000                          PRPAL   PROC    FAR
```

EJEMPLO 6–36 (página 2 de 4)

```
0000 B8 -- R              MOV   AX,DATA          ;cargar DS
0003 8E D8                MOV   DS,AX
0005 8E C0                MOV   ES,AX            ;cargar ES

0007 B4 09                MOV   AH,9             ; exhibir MES2
0009 BA 0017 R            MOV   DX,OFFSET MES2
000C CD 21                INT   21H

000E FC                   CLD                    ;seleccionar incremento automático
000F BF 0000 R            MOV   DI,OFFSET ARREG  ;direccionar ARREGLO
0012 B9 000A              MOV   CX,10            ;cargar conteo
0015             PRPAL1:

0015 E8 0039 R            CALL  LEE              ;leer número
0018 E2 FB                LOOP  PRPAL1           ;repetir 10 veces

001A BF 0000 R            MOV   DI,OFFSET ARREG  ;direccionar ARREGLO
001D B9 000A              MOV   CX,10            ;cargar contador
0020 E8 007A R            CALL  CLAS             ;clasificar números
0023 B4 09                MOV   AH,9             ;exhibir MES3
0025 BA 002D R            MOV   DX,OFFSET MES3
0028 CD 21                INT   21H

002A B9 000A              MOV   CX,10            ;cargar conteo
002D BE 0000 R            MOV   SI,OFFSET ARREG  ;direccionar ARREGLO

0030             PRPAL2:

0030 E8 009A R            CALL  DESP             ;exhibir número
0033 E2 FB                LOOP  PRPAL2

0035 B4 4C                MOV   AH,4CH           ;retornar a DOS
0037 CD 21                INT   21H

0039             PRPAL    ENDP

0039             LEE      PROC  NEAR

0039 33 ED                XOR   BP,BP            ;borrar el resultado
003B BB 000A              MOV   BX,10            ;cargar 10

003E             LEE1:

003E B4 06                MOV   AH,6             ;leer tecla
0040 B2 FF                MOV   DL,0FFH
0042 CD 21                INT   21H
0044 74 F8                JE    LEE1             ;si no hay tecla
0046 3C 0D                CMP   AL,13
0048 74 25                JE    LEE2             ;si es la tecla ENTRA
004A 3C 30                CMP   AL,'0'
004C 72 F0                JB    LEE1             ;si no es un número
004E 3C 39                CMP   AL,'9'
0050 77 EC                JA    LEE1             ;si no es un número

0052 50                   PUSH  AX
0053 8B C5                MOV   AX,BP            ;multiplicar por 10
0055 F7 E3                MUL   BX
0057 0B D2                OR    DX,DX            ;probar si es demasiado grande
0059 5A                   POP   DX
```

EJEMPLO 6–36 (página 3 de 4)

```
005A 75 E2                 JNZ    LEE1            ;demasiado grande
005C 52                    PUSH   DX
005D 32 F6                 XOR    DH,DH
005F 80 EA 30              SUB    DL,'0'          ;hacer BCD
0062 03 C2                 ADD    AX,DX           ;formar número
0064 5A                    POP    DX
0065 72 D7                 JC     LEE1            ;si es demasiado grande
0067 8B E8                 MOV    BP,AX           ;salvar el número
0069 B4 06                 MOV    AH,6            ;eco
006B CD 21                 INT    21H
006D EB CF                 JMP    LEE1
006F              LEE2:

006F 8B C5                 MOV    AX,BP
0071 AB                    STOSW
0072 B4 09                 MOV    AH,9            ;obtener línea nueva
0074 BA 0014 R             MOV    DX,OFFSET MES1
0077 CD 21                 INT    21H
0079 C3                    RET

007A              LEE      ENDP

007A              CLAS     PROC   NEAR

007A 49                    DEC    CX              ;ajustar conteo

007B              CLASI1:

007B 8B F7                 MOV    SI,DI           ;duplicar la dirección
007D 8B D9                 MOV    BX,CX           ;duplicar el contador
007F 33 ED                 XOR    BP,BP           ;borrar bandera de intercambio

0081              CLASI2:

0081 AD                    LODSW                  ;obtener datos
0082 3B 04                 CMP    AX,[SI]
0084 72 0A                 JB     SORT3           ;no hay intercambio
0086 8B 14                 MOV    DX,[SI]         ;hay intercambio
0088 89 04                 MOV    [SI],AX
008A 89 54 FE              MOV    [SI-2],DX
008D BD 0001               MOV    BP,1            ;indicar intercambio

0090              CLASI3:

0090 4B                    DEC    BX
0091 75 EE                 JNE    SORT2           ;repetir
0093 0B ED                 OR     BP,BP
0095 74 02                 JE     SORT4           ;hecho
0097 E2 E2                 LOOP   SORT1           ;repetir

0099              CLASI4:

0099 C3                    RET

009A              CLASI    ENDP
009A              DESP     PROC   NEAR

009A 51                    PUSH   CX              ;salvar conteo externo
009B 33 C9                 XOR    CX,CX           ;borrar conteo
```

EJEMPLO 6–36 (página 4 de 4)

```
009D BB 000A                MOV   BX,10            ;cargar 10
00A0 AD                     LODSW                  ;obtener número

00A1               DESP1:

00A1 33 D2                  XOR   DX,DX            ;borrar DX
00A3 F7 F3                  DIV   BX               ;dividir entre 10
00A5 52                     PUSH  DX               ;salvar el residuo
00A6 41                     INC   CX
00A7 0B C0                  OR    AX,AX            ;probar el cociente
00A9 75 F6                  JNZ   DESP1            ;repetir
00AB B4 06                  MOV   AH,6

00AD               DESP2:

00AD 5A                     POP   DX               ;obtener el digito
00AE 80 C2 30               ADD   DL,'0'           ;hacerlo ASCII
00B1 CD 21                  INT   21H              ;exhibirlo
00B3 E2 F8                  LOOP  DESP2            ;repetir

00B5 B4 09                  MOV   AH,9             ;obtener nueva linea
00B7 BA 0014 R              MOV   DX,OFFSET MES1
00BA CD 21                  INT   21H

00BC 59                     POP   CX               ;restaurar CX
00BD C3                     RET

00BE               DISP     ENDP

00BE               CODE     ENDS

                            END   PRPAL
```

Una vez clasificados los números, se exhiben en la pantalla en orden numérico ascendente. No se han previsto errores cuando se teclea cada número. El programa termina después de clasificar un grupo de 10 números y hay que volver a llamarlo para clasificar 10 nuevos números.

Expansión de archivo hexadecimal

Un ejemplo de un programa que exhibe un archivo en forma hexadecimal, permite practicar acceso al disco. También da la oportunidad de leer un parámetro (nombre del archivo) de la línea de comando de DOS.

Siempre que se teclea un comando (nombre del programa) en la línea de orden de DOS, cualesquiera parámetros que sigan se colocan en un *prefijo de segmento de programa.* Ese prefijo (PSP) se presenta en la figura A-6 del apéndice A. Se verá que la longitud de la línea del comando y de los parámetros de la línea de comando aparecen en el PSP junto con otra información. Cuando se ejecuta el programa, el registro de segmento DS direcciona al PSP, por lo cual se utiliza una dirección de desplazamiento de 80H para accesar a la longitud (tamaño byte) de la línea de comando. Una vez obtenida la longitud, se puede accesar a la línea de comando y a sus parámetros.

EJEMPLO 6-37 (página 1 de 5)

```
0000                       STAC    SEGMENT STACK

0000 0400[                         DW      1024 DUP (?)       ;inicializar la pila
              ????
                  ]

0800                       STAC    ENDS

0000                       DATOS   SEGMENT

0000 0100[                         BUFFER  DB   256 DUP (?)
              ??
                 ]
0100 0040[                         ARCH    DB   64 DUP (?)
              ??
                 ]

0140 0D 0A 2A 2A 2A 20             MES1    DB   13,10,'***Usted da nombre al archivo *** ',13,10,'$'
     59 6F 75 20 65 6E
     74 65 72 20 66 69
     6C 65 20 6E 61 6D
     65 20 2A 2A 2A 0D
     0A 24
0160 0D 0A 2A 2A 2A 20             MES2    DB   13,10,'****No se encontró el archivo*** '13,10,'$'
     46 69 6C 65 20 6E
     6F 74 20 66 6F 75
     6E 64 20 2A 2A 2A
     0D 0A 24
017B 0D 0A 2A 2A 2A 20             MES3    DB   13,10,'***Archivo corrupto*** ',13,10,'$'
     46 69 6C 65 20 63
     6F 72 72 75 70 74
     20 2A 2A 2A 0D 0A
     24
0194 0D 0A 53 65 63 74             MES4    DB   13,10,'Sección: $'
     69 6F 6E 3A 20 24
01A0 0D 0A 24                      MES5    DB   13,10,'$'
01A3 00                            SECT    DB   ?

01A4                       DATA    ENDS

0000                       CODE    SEGMENT

                                   ASSUME CS:CODE,DS:DATA,SS:STAC

0000                       PRPAL   PROC    FAR

0000 B8 -- R                       MOV     AX,DATA            ;cargar ES
0003 8E C0                         MOV     ES,AX
0005 BE 0082                       MOV     SI,82H             ;longitud de la dirección
0008 80 7C FE 00                   CMP     BYTE PTR [SI-2],0
000C 75 0D                         JNE     PRPAL2             ;si es el nombre del archivo

000E 8E D8                         MOV     DS,AX              ;cargar DS
0010 BA 0140 R                     MOV     DX,OFFSET MES1     ;exhibir error

0013                       PRPAL1:

0013 B4 09                         MOV     AH,9
0015 CD 21                         INT     21H

0017 B4 4C                         MOV     AH,4CH             ;retornar a DOS
```

EJEMPLO 6–37 (página 2 de 5)

```
0019 CD 21                  INT     21H

001B               PRPAL2:

001B 8A 4C FE               MOV     CL,[SI-2]         ;obtener longitud
001E 32 ED                  XOR     CH,CH             ;hacerlo de 16 bits
0020 49                     DEC     CX                ;ajustar contador
0021 BF 0100 R              MOV     DI,OFFSET ARCH    ;direccionar nombre de archivo
0024 F3/ A4                 REP MOVSB                 ;salvar nombre del archivo
0026 C6 05 00               MOV     BYTE PTR [DI],0   ;hacer cadena de ASCII-Z
0029 8C C0                  MOV     AX,ES             ;cargar DS
002B 8E D8                  MOV     DS,AX

002D B8 3D02                MOV     AX,3D02H          ;abrir archivo
0030 BA 0100 R              MOV     DX,OFFSET ARCH
0033 CD 21                  INT     21H
0035 73 05                  JNC     PRPAL3            ;si se encuentra el ARCHIVO
0037 BA 0160 R              MOV     DX,OFFSET MES2    ;exhibir no se encontró el archivo
003A EB D7                  JMP     PRPAL1

003C               PRPAL3:

003C 8B D8                  MOV     BX,AX             ;obtener manejador de archivo
003E C6 06 01A3 R FF        MOV     SECT,-1           ;desarrollar primera sección

0043               PRPAL4:

0043 FE 06 01A3 R           INC     SECT
0047 B4 3F                  MOV     AH,3FH            ;leer 256 bytes
0049 B9 0100                MOV     CX,256
004C BA 0000 R              MOV     DX,OFFSET BUFFER
004F CD 21                  INT     21H
0051 73 05                  JNC     PRPAL5            ;si la lectura fue buena
0053 BA 017B R              MOV     DX,OFFSET MES3    ;exhibir archivo corrupto
0056 EB BB                  JMP     PRPAL1
0058               PRPAL5:

0058 3D 0000                CMP     AX,0              ;probar si es el final del archivo
005B 75 0B                  JNE     PRPAL6            ;no es el final

005D B4 3E                  MOV     AH,3EH            ;cerrar el archivo
005F CD 21                  INT     21H

0061 E8 00BF R              CALL    NUEVO

0064 B4 4C                  MOV     AH,4CH            ;salir a DOS
0066 CD 21                  INT     21H

0068               PRPAL6:

0068 E8 006D R              CALL    DUMP              ;exhibir sección de 256 bytes
006B EB D6                  JMP     MAIN4             ;repetir

006D               PRPAL    ENDP

006D               DUMP     PROC    NEAR              ;exhibir sección de 256 bytes

006D 8B C8                  MOV     CX,AX             ;salvar conteo
006F B4 09                  MOV     AH,9              ;exhibir el encabezado
```

EJEMPLO 6–37 (página 3 de 5)

```
0071 BA 0194 R              MOV    DX,OFFSET MES4
0074 CD 21                  INT    21H

0076 A0 01A3 R              MOV    AL,SECT          ;exhibir el número
0079 E8 009C R              CALL   DISP

007C BE 0000 R              MOV    SI,OFFSET BUFFER

007F                 EXPAN1:
007F E8 00C7 R       CALL   DESPD                   ;exhibir dirección

0082                 EXPAN1A:

0082 E8 00FA R              CALL   DESPN            ;exhibir número
0085 49                     DEC    CX
0086 74 08                  JE     DUMP2
0088 8B C6                  MOV    AX,SI
008A 24 0F                  AND    AL,0FH           ;probar dirección
008C 74 F1                  JZ     EXPAN1
008E EB F2                  JMP    EXPAN1A
0090                 EXPAN2:

0090 B4 06                  MOV    AH,6             ;obtener una tecla
0092 B2 FF                  MOV    DL,0FFH
0094 CD 21                  INT    21H
0096 74 F8                  JE     EXPAN2
0098 E8 00BF R              CALL   NUEVO
009B C3                     RET

009C                 EXPAN  ENDP

009C                 DESP   PROC   NEAR             ;exhibir decimal

009C 51                     PUSH   CX               ;salvar CX
009D 53                     PUSH   BX               ;salvar BX
009E 32 E4                  XOR    AH,AH
00A0 BB 000A                MOV    BX,10            ;cargar 10
00A3 33 C9                  XOR    CX,CX            ;borrar contador

00A5                 DESP1:

00A5 33 D2                  XOR    DX,DX
00A7 F7 F3                  DIV    BX               ;dividir entre 10
00A9 52                     PUSH   DX               ;salvar el residuo
00AA 41                     INC    CX               ;contar el residuo
00AB 0B C0                  OR     AX,AX            ;probar el cociente
00AD 75 F6                  JNE    DESP1            ;repetir
00AF B4 06                  MOV    AH,6

00B1                 DESP2:

00B1 5A                     POP    DX               ;obtener dígito
00B2 80 C2 30               ADD    DL,'0'           ;convertir a ASCII
00B5 CD 21                  INT    21H
00B7 E2 F8                  LOOP   DESP2            ;repetir
00B9 E8 00BF R              CALL   NUEVO            ;obtener nueva linea
00BC 5B                     POP    BX
00BD 59                     POP    CX
00BE C3                     RET
```

EJEMPLO 6–37 (página 4 de 5)

```
00BF                    DESP    ENDP

00BF                    NUEVO   PROC    NEAR            ;exhibir nueva línea

00BF B4 09                      MOV     AH,9
00C1 BA 01A0 R                  MOV     DX,OFFSET MES5
00C4 CD 21                      INT     21H
00C6 C3                         RET
00C7                    NUEVO   ENDP
00C7                    DESPD   PROC    NEAR            ;exhibir dirección

00C7 E8 00BF R                  CALL    NUEVO           ;obtener nueva línea
00CA 8B C6                      MOV     AX,SI           ;obtener dirección
00CC E8 00DF R                  CALL    DIG             ;exhibir dígito
00CF E8 00DF R                  CALL    DIG
00D2 E8 00DF R                  CALL    DIG
00D5 E8 00DF R                  CALL    DIG

00D8 B4 06                      MOV     AH,6            ;exhibir espacio
00DA B2 20                      MOV     DL,''
00DC CD 21                      INT     21H

00DE C3                         RET

00DF                    DESPD   ENDP

00DF                    DIG     PROC    NEAR            ;exhibir dígito hexadecimal

00DF D1 C0                      ROL     AX,1            ;ubicar dígito
00E1 D1 C0                      ROL     AX,1
00E3 D1 C0                      ROL     AX,1
00E5 D1 C0                      ROL     AX,1
00E7 50                         PUSH    AX
00E8 24 OF                      AND     AL,0FH          ;enmascarar
00EA 04 30                      ADD     AL,'0'          ;convertir a ASCII
00EC 3C 39                      CMP     AL'9'
00EE 76 02                      JBE     DIG1
00F0 04 07                      ADD     AL,7

00F2                    DIG1:

00F2 8A D0                      MOV     DL,AL           ;exhibir dígito
00F4 B4 06                      MOV     AH,6
00F6 CD 21                      INT     21H
00F8 58                         POP     AX
00F9 C3                         RET

00FA                    DIG     ENDP

00FA                    DESPN   PROC    NEAR            ;exhibir número

00FA AC                         LODSB                   ;obtener número

00FB 8A E0                      MOV     AH,AL
00FD E8 00DF R                  CALL    DIG             ;exhibir dígito
0100 E8 00DF R                  CALL    DIG

0103 B4 06                      MOV     AH,6
```

EJEMPLO 6–37 (página 5 de 5)

```
0105 B2 20                MOV   DL,' '          ;exhibir espacio
0107 CD 21                INT   21H
0109 C3                   RET

010A              DESPN   ENDP

010A              CODE    ENDS

                          END   PRPAL
```

En el ejemplo 6-37 se lista un programa que obtiene un nombre de archivo de la línea de comando y, luego, exhibe el archivo en forma hexadecimal. Este programa es útil para depurar programas con errores y también para practicar accceso a archivos de disco y conversiones. El parámetro que sigue al comando siempre empieza con un espacio (20H) en la dirección desplazada 81H y siempre termina con un retorno de carro (0DH). La longitud del parámetro es siempre mayor por uno. Por ejemplo, si se teclea EXPAN RANA en la línea de comando y EXPAN es el nombre del programa, el parámetro RANA se almacena comenzando con un 20H en el desplazamiento 81H y la longitud es 5.

6–6 CAMBIO DE LOS SERVICIOS DE INTERRUPCION

Los cambios a los servicios de interrupción se utilizan para cambiar la estructura de interrupción del microprocesador. Por ejemplo, se puede cambiar el servicio de la interrupción del teclado a fin de poder detectar una tecla de acción especial llamada tecla "caliente". Siempre que se oprime la tecla de acción especial, se puede accesar a un programa del tipo termina y permanece residente (TSR) que efectúa una tarea especial. Algunos de los ejemplos de este tipo de programa software de tecla "caliente" son calculadoras, relojes y otros.

Una interrupción

Para poder cambiar el servicio de interrupción, se debe utilizar una llamada a una función del DOS que lea la dirección actual del vector de interrupción. Se utiliza la llamada a la función 35H del DOS para leer el vector de interrupción actual y la función 25H del DOS se utiliza para cambiar la dirección del vector actual. En ambas funciones el DOS AL indica el número (00H a FFH) del vector y AH indica el número de llamada a la función del DOS.

Cuando se lee el vector con la función 35H, la dirección del desplazamiento retorna al registro BX y la dirección del segmento está en el registro ES. Estos dos registros se salvan para poder recuperarlos cuando se elimine de la memoria el cambio del servicio de la interrupción. El vector, se inicializa a la dirección almacenada en la localidad de memoria direccionada por DS:DX.

EJEMPLO 6–38

```
                         ORG     100H                          ;origen de un programa COM

0100                     INICIO

0100 EB 04                       JMP     MAIN

0102 00000000            DIRECC  DD      ?                     ;vector de interrupción viejo

0106                     PRPAL:

0106 8C C8                       MOV     AX,CS                 ;direccionar CS con DS
0108 8E D8                       MOV     DS,AX

                     ;obtiene dirección del vector 0

010A B8 3500             MOV             AX,3500H
010D CD 21               INT             21H
                     ;salvar dirección del vector

010F 2E: 89 1E 0102 R    MOV     WORD PTR DIRECC,BX
0114 2E: 8C 06 0104 R    MOV     WORD PTR DIRECC+2,ES

                     ;instalar nueva dirección del vector de interrupción 0

0119 B8 2500             MOV     AX,2500H
011C BA 0300 R           MOV     DX,OFFSET NUEVO
011F CD 21               INT     21H
```

El proceso de instalación de un manejador de interrupción por medio de un cambio en el servicio de la interrupción se muestra en el ejemplo 6-38. Con este procedimiento se lee la dirección en curso del vector de interrupción y se almacena en una localidad de memoria de doble palabra para accesar el nuevo procedimiento de servicio de la interrupción. Luego, la dirección del nuevo procedimiento de servicio de la interrupción almacenada en DS:DX se carga en el vector con el empleo función 25H del DOS.

Ejemplo de reloj TSR para alarma

Un ejemplo muy sencillo del cambio del servicio de una interrupción y un programa TSR es producir un "bip" en la bocina una vez que transcurre cierto tiempo. La cantidad de tiempo se especifica a intervalos de 10 segundos definidos por un directivo igualar que se encuentra en el programa. Por ejemplo, si la duración del intervalo especificado es de 60, entonces ocurrirá el "bip" (alarma) después de 10 minutos una vez instalado el programa. Para ajustar el intervalo de temporización del "bip" se cambia el número almacenado con el directivo igualar.

EJEMPLO 6–39 (página 1 de 3)

```
0000                     CODES   SEGMENT 'CODE'

                                 ASSUME  CS:CODES

                                 ORG     5DH
```

EJEMPLO 6–39 (página 2 de 3)

```
005D 0000           CUENTA  DW      ?               ;intervalo del temporizador

                            ORG     100H            ;origen para un programa COM

0100                INICIO:

0100 E9 0091                JMP     PRPAL

0103 00000000       DIREC1  DD      ?               ;vector de interrupción viejo
0107 B6             SEG10   DB      182             ;contador de 10 segundos
0108 00             ACTIVO  DB      0               ;bandera del temporizador

= 000A              ALARMA  EQU     10              ;alarma ocurre cada 10 segundos
= 0320              TONO    EQU     800             ;frecuencia en hertz
= 0006              LONGI   EQU     6               ;duración = longitud/18.2

0109                BIP     PROC    NEAR

0109 50                     PUSH    AX              ;salvar registros
010A 53                     PUSH    BX
010B 51                     PUSH    CX
010C 52                     PUSH    DX
010D 06                     PUSH    ES
010E B0 B6                  MOV     AL,0B6H         ;inicializar temporizador 2
0110 E6 43                  OUT     43H,AL

0112 BA 0012                MOV     DX,12H          ;calcular el conteo
0115 B8 34DC                MOV     AX,34DCH
0118 BB 0320                MOV     BX,TONE
011B F7 F3                  DIV     BX

011D E6 42                  OUT     42H,AL          ;programar temporizador 2
011F 8A C4                  MOV     AL,AH
0121 E6 42                  OUT     42H,AL

0123 E4 61                  IN      AL,61H          ;bocina funciona
0125 0C 03                  OR      AL,3
0127 E6 61                  OUT     61H,AL

0129 BA 0006                MOV     DX,LENG
012C 2B C9                  SUB     CX,CX
012E 8E C1                  MOV     ES,CX
0130 26: 03 16 046C         ADD     DX,ES:[46CH]
0135 26: 13 0E 046E         ADC     CX,ES:[46EH]
013A                BEEPS:

013A 26: 8B 1E 046C         MOV     BX,ES:[46CH]    ;espera longitud pulsos de reloj
013F 26: A1 046E            MOV     AX,ES:[46EH]
0143 2B DA                  SUB     BX,DX
0145 1B C1                  SBB     AX,CX
0147 72 F1                  JC      BEEPS
0149 E4 61                  IN      AL,61H          ;bocina apagada
014B 34 03                  XOR     AL,3
014D E6 61                  OUT     61H,AL

014F 07                     POP     ES              ;restaurar registros
0150 5A                     POP     DX
0151 59                     POP     CX
```

EJEMPLO 6–39 (página 3 de 3)

```
0152 5B                         POP     BX
0153 58                         POP     AX
0154 C3                         RET

0155                    BEEP    ENDP

0155                    TIMES   PROC    FAR                     ;interrupción nueva

0155 2E: 80 3E 0108 R           CMP     ACTIVE,0                ;probar activo
     00
015B 74 05                      JZ      TIMES1                  ;si está armado
015D 2E: FF 2E 0103 R           JMP     ADDR1                   ;hacer interrupción original

0162                    TIMES1:
0162 2E: FE 06 0108 R           INC     ACTIVE                  ;inicializar activo
0167 9C                         PUSHF                           ;simular interrupción
0168 2E: FF 1E 0103 R           CALL    ADDR1                   ;hacer interrupción
016D FB                         STI                             ;habilitar INTR
016E 1E                         PUSH    DS                      ;salvar registros
016F 0E                         PUSH    CS
0170 1F                         POP     DS                      ;obtener segmento de código actual
0171 2E: FE 0E 0107 R           DEC     SEC10                   ;decrementar contador
0176 75 15                      JNZ     TIMES2                  ;si no es de 10 segundos
0178 2E: C6 06 0107 R           MOV     SEC10,182               ;volver a cargar SEG10
     B6
017E 2E: FF 0E 005D R           DEC     CUENTA                  ;decrementar contador
0183 75 08                      JNZ     TIEMPO2                 ;si no hay alarma
0185 E8 FF81                    CALL    BEEP                    ;hacer sonar la alarma
0188 2E: FE 06 0108 R           INC     ACTIVO

018D                    TIEMPO2
018D 2E: FE 0E 0108 R           DEC     ACTIVO                  ;decrrementar activo
0192 1F                         POP     DS
0193 CF                         IRET

0194                    TIEMPO  ENDP
0194                    PRPAL:
0194 8C C8                      MOV     AX,CS
0196 8E D8                      MOV     DS,AX

0198 B8 000A                    MOV     AX,ALARMA
019B 2E: A3 005D R              MOV     CUENTA,AX               ;salvar intervalo de alarma
019F B8 3508                    MOV     AX,3508H                ;obtener vector 8
01A2 CD 21                      INT     21H
01A4 2E: 89 1E 0103 R           MOV     WORD PTR DIREC1,BX      ;salvar dirección
01A9 2E: 8C 06 0105 R           MOV     WORD PTR DIREC1+2,ES
01AE B8 2508                    MOV     AX,2508H                ;instalar nueva interrupción
01B1 BA 0155 R                  MOV     DX,OFFSET TIEMPO
01B4 CD 21                      INT     21H
01B6 BA 0194 R                  MOV     DX,OFFSET PRPAL         ;desarrollar la longitud (duración)
01B9 B1 04                      MOV     CL,4
01BB D3 EA                      SHR     DX,CL
01BD 42                         INC     DX
01BE B8 3100                    MOV     AX,3100H                ;TSR
01C1 CD 21                      INT     21H

01C3                    CODES   ENDS

                                END     INICIO
```

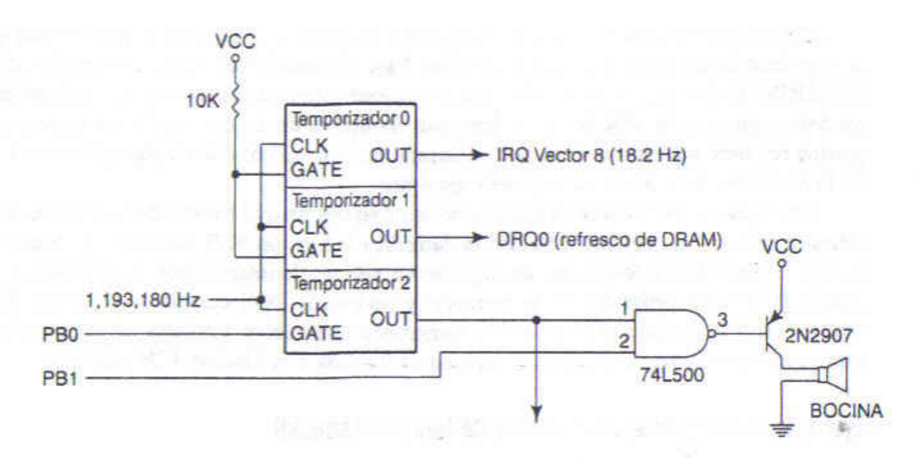


FIGURA 6-7 El circuito de la bocina conectado con el temporizador de la computadora personal. (El 8255 está en los puertos de E/S 60H a 63H y el 8253 está en los puertos de E/S 40H a 43H.)

El "bip" o alarma se produce con el empleo del temporizador 2 de la PC, a fin de generar un sonido audible en la bocina. (Consulte la sección 9-5 para la descripción del temporizador y la figura 6-7 para la conexión en la computadora.) La programación del temporizador 2 con una frecuencia o tono particulares del "bip" se logra al programarlo con 1 193 180 dividido entre el tono deseado. Por ejemplo, si se divide ese 1 193 180 entre 800, se produce en la bocina un tono de audio de 800 Hz. Consulte el procedimiento "BEEP" (ejemplo 6-39) para la programación del temporizador y encender y apagar la bocina después de una breve espera determinada por el número de pulsos del reloj. En este procedimiento se emplean 6 pulsos de reloj que producen un bip de alrededor de 1/3 de segundo. Se debe tener en cuenta que cada pulso de reloj ocurre alrededor de 18.2 segundos (el tiempo real es 18.2064819336). Para lograrlo se emplean las localidades de espera del temporizador para el usuario en el primer segmento de la memoria. El temporizador de espera para el usuario se actualiza 18.2 veces por segundo en la computadora, por lo cual se puede utilizar para temporizar eventos.

El programa PRINCIPAL inicializa el contador de la alarma con un 10 que ocasiona una demora de 100 segundos antes que se produzca el bip en la bocina. Una vez inicializado el retardo de la alarma, se obtiene la dirección del vector original (tipo 8) de la interrupción del temporizador con el empleo de la función 35H del DOS y se almacena en la localidad DI1 de la memoria. Una vez obtenido el vector original de la interrupción, se emplea la función 25H del DOS para almacenar la dirección de VECES que es el nuevo procedimiento del servicio de la interrupción, en el vector número 8. Las últimas instrucciones del programa principal se utilizan para hacer que el procedimiento del servicio de la interrupción y del BIP residan en la memoria. Para ello, primero se calcula la longitud de estos procedimientos. Se debe tener en cuenta que la longitud se expresa con el número de párrafos de 16 bytes. Cuando se ejecuta la llamada a la función 31H del DOS, el nuevo procedimiento del servicio de la interrupción y del BIP se vuelven residentes en la memoria. No hay ninguna provisión para retirar el software residente de la memoria del sistema una vez que quedó instalado. Todo lo que ocurre después de que suena la alarma, es que se "desactiva" el procedimiento de servicio de la interrupción.

El programa se ensambla y se encadena con cualquier otro programa, pero en este ejemplo se le convierte de un archivo EXE a uno COM. Para efectuar la conversión, se emplea el programa EXE2BIN. Todos los archivos COM deben empezar a ejecutar el software en la localidad 100H y no deben exceder de 64K bytes de longitud. Se utiliza un archivo COM en lugar de uno EXE porque requiere menos memoria para almacenarlo. Esto es importante para programas residentes en la memoria, pero no es un requisito absoluto.

Este ejemplo permanece residente incluso después de que ha sonado la alarma. Si se quiere retirar el programa, debe terminar con la llamada a la función 4CH del DOS, tal como se termina la mayor parte de los programas en lugar de un retorno de interrupción. La llamada a la función del DOS retira el programa de la memoria y retorna al DOS cuando se le activa. Si se desea terminar con la función 4CH de DOS, compruebe que vuelve a colocar el vector original de la interrupción en el vector 8 antes de ejecutar la llamada a la función 4CH del DOS.

Ejemplo de programa con la tecla de función especial

Las teclas de función especial ("calientes") se oprimen para llamar programas que van a terminar y a permanecer residentes. Por ejemplo, se podría definir que la combinación de teclas ALT C es una tecla de función especial que llama a un programa que exhibe la hora. Se debe tener en cuenta que la tecla de función especial se detecta dentro de la mayor parte de las aplicaciones, pero no en la línea de comandos del DOS, en donde, si se utilizase, podría cerrar el sistema. Para detectar una tecla de función especial, por lo general se cambia el vector de la interrupción 9, que es la interrupción del teclado que ocurre si se oprime cualquier tecla. Esto permite probar el teclado y detectar una tecla de función especial antes que la interrupción normal haga el procesamiento de la tecla oprimida.

EJEMPLO 6–40

```
0108 00000000          VIE9   DD     ?             ;dirección original del vector
010C 00                CODI   DB     ?             ;código de rastreo de la tecla 'caliente'
010D 00                MASC   DB     ?             ;cambio a mayúscula y Alt
010E 00                CAL    DB     ?             ;corregir cambio 'caliente'/alternativa
010F 00                BAND   DB     0             ;bandera de TSR

0200 FB                VEC9:  STI                  ;interrupción conectada
0201 50                       PUSH   AX
0202 E4 60                    IN     AL,60H        ;obtener código de rastreo
0204 2E: 3A 06 010C R         CMP    AL,CODI       ;verificar código de rastreo de la tecla 'caliente'
0209 75 2E                    JNE    VEC91         ;si no es tecla caliente
020B 06                       PUSH   ES
020C 2B C0                    SUB    AX,AX
020E 8E C0                    MOV    ES,AX
0210 26: A0 0417              MOV    AL,ES:[417H]  ;obtener estado de cambio a mayúsculas y Alt
0214 2E: 22 06 010D R         AND    AL,MASC       ;aislar el o los bits
0219 2E: 3A 06 010E R         CMP    AL,CAL        ;probar cambio a mayúsculas y Alt
021E 07                       POP    ES
021F 75 18                    JNE    VEC91         ;si no es tecla 'caliente'

                 ¡No se presionó tecla 'caliente'!

0221FA                        CLI                  ;interrupción apagada
0222 E4 61                    IN     AL,61H        ;desechar 'teclazo'
0224 0C 80                    OR     AL,80H        ;borrar teclado
```

```
0226 E6 61                    OUT    61H,AL
0228 24 7F                    AND    AL,7FH        ;suspensión de señal
022A E6 61                    OUT    61H,AL
022C B0 20                    MOV    AL,20H        ;inicializar controlador de interrupciones
022E E6 20                    OUT    20H,AL
0230 FB                       STI                  ;interrupciones conectadas
0231 58                       POP    AX
0232 2E: C6 06 010F R         MOV    BAND,1        ;indicar tecla 'caliente'
     01
0238 CF                       IRET                 ;salir del manejador

                   ;no se presionó tecla caliente

0239                    VEC91:

0239 58                       POP    AX
023A FA                       CLI                  ;interrupciones deshabilitadas
023B 9C                       PUSHF                ;simular interrupción
023C 2E: FF 1E 0108 R         CALL   VIE9          ;llamar interrupción original del teclado
0241 CF                       IRET
```

En el ejemplo 6-40 se muestra un procedimiento de cambio corto para el servicio de la interrupción y que prueba si hay una tecla de función especial y activa una bandera si la detecta. En este ejemplo no se ilustra el programa para instalar el procedimiento de reemplazo. La variable CODI es el código de rastreo (figura 6-8) de la tecla de función especial que se va a detectar, las MASCS son los bits de máscara para extraer las teclas de cambio Alt de la memoria en la

Esc	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	Prt	Scr	Pse
01	3B	3C	3D	3E	3F	40	41	42	43	44	57	58	⇔	57	§

'	1	2	3	4	5	6	7	8	9	0	–	=	←	Ins	Hom	PgU	Num	/	*	–
29	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	•52	•47	•49	45	•36	37	4A

Tab	Q	W	E	R	T	Y	U	I	O	P	[	]	\	Del	End	PgD	7	8	9	+
0F	10	11	12	13	14	15	16	17	18	19	1A	1B	2B	•53	•4F	•51	47	48	49	4E

Caps	A	S	D	F	G	H	J	K	L	;	'	↵	4	5	6
3A	1E	1F	20	21	22	23	24	25	26	27	28	1C	4B	4C	4D

Shift	Z	X	C	V	B	N	M	,	.	/	Shift	↑	1	2	3	↵
2A	2C	2D	2E	2F	30	31	32	33	34	35	36	48	4F	50	51	1C

Ctrl	Alt	Barra espaciadora	Alt	Ctrl	←	↓	→	Ins	Del
1D	38	39	38	10	4B	50	40	52	53

NOTAS
• = E0 + código de rastreo
⇔ = E0 2A E0 37
§ = E0 10 45

FIGURA 6-8 Los códigos de rastreo del teclado.

localidad 0000:0471. (Consulte los detalles en el apéndice A) y el código CAL es la combinación deseada de teclas de cambio y Alt para la tecla de función especial. Con este procedimiento se detecta una tecla "caliente" y si está en funcionamiento, el procedimiento coloca un 01H en el byte BANDERA, el cual se suele detectar con un cambio en el vector de la interrupción 8.

En el ejemplo 6-41 se presenta un programa que cambia los vectores de interrupción 9 y 8. El vector 9 se captura para accesar a una tecla "caliente" y el vector 8 se captura para ejecutar un programa TSR que se activa con la tecla "caliente". En este ejemplo se emplea la tecla "caliente" A que accesa a un programa TSR para exhibir la hora en la pantalla. La hora se exhibe hasta que se oprime la tecla de escape (Esc) en el teclado. Cuando se oprime Esc, los datos originales se vuelven a exhibir por encima de la hora. Este programa sólo funciona si el adaptador para la pantalla está en modo texto.

EJEMPLO 6–41 (página 1 de 5)

```
                          .286
                          ;
                          ;Este programa exhibe la hora cuando se oprimen la tecla Alt y T
                          ;Este programa se establece como archivo .COM con ORG 100H
                          ;
0000                      CODE   SEGMENT

                                 ASSUME  CS:CODE

                                 ORG     100H

0100 E9 0962              BEGIN: JMP     START

0103 00000000             VEC8    DD     ?              ;dirección original del vector 8
0107 00000000             VEC9    DD     ?              ;dirección original del vector 9
010B 00000000             STPT    DD     ?              ;apuntador original de la pila
010F 00                   H_FLAG  DB     0              ;bandera de tecla 'caliente'
0110 14                   H_CODE  DB     14H            ;código de rastreo para tecla 'caliente', ''T''
0111 08                   MASKS   DB     8              ;máscara alterna
0112 08                   HOT     DB     8              ;'caliente' correcta cambio/Alt
0113 00                   COL     DB     0              ;exhibir columna
0114 00                   REN     DB     0              ;exhibir renglón
0115 00                   T_REN   DB     ?              ;renglón temp
0116 00                   T_COL   DB     ?              ;columna temp
0117 0005 [               BUF     DB     5 DUP (?)      ;dato de exhibición viejo
           00
              ]
011C 24                           DB     '$'
011D 0005 [               BUF1    DB     5 DUP (?)      ;memoria (buffer) de tiempo
           00
              ]
0122 24                           DB     '$'
0123 0000                 TEMP    DW     ?
0125 0400 [                       DW     400H DUP (?)
           0000
                ]
0925 = 0925               STAC    EQU    THIS WORD

                          ;Se captó tecla 'caliente'!

0925                      TECLA   PROC   FAR
```

EJEMPLO 6–41 (página 2 de 5)

```
0925 FB                          STI                    ;interrupciones habilitadas
0926 50                          PUSH  AX
0927 E4 60                       IN    AL,60H           ;obtener código de rastreo
0929 2E: 3A 06 0110 R            CMP   AL,H_CODE        ;probar si es código de rastreo de tecla 'caliente'
092E 75 2E                       JNE   TECLA1           ;si no fue tecla caliente
0930 06                          PUSH  ES
0931 2B C0                       SUB   AX,AX
0933 8E C0                       MOV   ES,AX
0935 26: A0 0417                 MOV   AL,ES:[417H]     ;obtener estado de tecla de cambio/ Alt
0939 2E: 22 06 0111 R            AND   AL,MASCS         ;aislar tecla alterna
093E 2E: 3A 06 0112 R            CMP   AL,CALI          ;probar si hay alterna
0943 07                          POP   ES
0944 75 18                       JNE   TECLA1           ;si no hay tecla caliente

                   ;Si se detectó tecla 'caliente'!

0946 FA                          CLI                    ;interrupciones deshabilitadas
0947 E4 61                       IN    AL,61H           ;desechar 'teclazo'
0949 0C 80                       OR    AL,80H           ;borrar el teclado
094B E6 61                       OUT   61H,AL
094D 24 7F                       AND   AL,7FH           ;suspensión de señal
094F E6 61                       OUT   61H,AL
0951 B0 20                       MOV   AL,20H           ;reinicializar controlador de interrupciones
0953 E6 20                       OUT   20H,AL
0955 FB                          STI                    ;interrupciones habilitadas
0956 58                          POP   AX
0957 2E: C6 06 010F R            MOV   C_BAN,0FFH       ;indicar tecla 'caliente'
     FF
095D CF                          IRET

                   ;Si no se detectó tecla 'caliente'!

095E               TECLA1:

095E 58                          POP   AX
095F FA                          CLI                    ;interrupciones deshabilitadas
0960 9C                          PUSHF                  ;simular interrupción
0961 2E: FF 1E 0107 R            CALL  CS:VEC9
0966 CF                          IRET

0967               TECLA         ENDP

                   ;interrupción pata TICK de reloj!

0967               TICK          PROC  FAR

0967 2E: 80 3E 010F R            CMP   C_BAN,0          ;probar si hay tecla 'caliente'
     00
096D 75 05                       JNZ   TICK1            ;si tecla 'caliente' está activa
096F 2E: FF 2E 0103 R            JMP   CS:VEC8          ;hacer interrupción normal

                   ;Si está activa tecla 'caliente'!

0974               TICK1:

0974 2E: C6 06 010F R            MOV   C_BAN,0          ;borrar solicitud de tecla 'caliente'
     00
097A 9C                          PUSHF                  ;hacer interrupción por impulso de reloj
097B 2E: FF 1E 0103 R CALL       CS:VEC8
```

EJEMPLO 6–41 (página 3 de 5)

```
0980 FA                        CLI
0981 2E: A3 0123 R             MOV    TEMP,AX              ;salvar zona de pila vieja
0985 8C D0                     MOV    AX,SS
0987 2E: A3 010D R             MOV    WORD PTR STPT+2,AX
098B 2E: 89 26 010B R          MOV    WORK PTR STPT,SP

0990 BC 0925 R                 MOV    SP,OFFSET STAC       ;obtener zona de pila nueva
0993 8C C8                     MOV    AX,CS
0995 8E D0                     MOV    SS,AX
0997 FB                        STI

0998 E8 0012                   CALL   RELOJ                ;exhibir hora hasta que se oprima alguna tecla

099B FA                        CLI
099C 2E: A1 010D R             MOV    AX,WORD PTR STPT+2;direccionar zona de pila vieja
09A0 8E D0                     MOV    SS,AX
09A2 2E: 8B 26 010B R          MOV    SP,WORD PTR STPT
09A7 2E: A1 0123 R             MOV    AX,TEMP
09AB FB                        STI

09AC CF                        IRET

09AD                    TICK   ENDP

                        ;procedimiento RELOJ para exhibir la hora en COL, RENGLON

09AD                    RELOJ  PROC   NEAR


09AD 50                        PUSH   AX                   ;salvar los registros
09AE 53                        PUSH   BX
09AF 51                        PUSH   CX
09B0 52                        PUSH   DX
09B1 57                        PUSH   DI
09B2 1E                        PUSH   DS
09B3 8C C8                     MOV    AX,CS
09B5 8E D8                     MOV    DS,AX

09B7 B4 03                     MOV    AH,3                 ;obtener posición actual del cursor
09B9 B7 00                     MOV    BH,0
09BB CD 10                     INT    10H
09BD 2E: 88 36 0115 R          MOV    T_REN,DH             ;salvar renglón y columna
09C2 2E: 88 16 0116 R          MOV    T_COL,DL
09C7 B4 02                     MOV    AH,2                 ;mover el cursor
09C9 2E: 8A 36 0114 R          MOV    DH,REN
09CE 2E: 8A 16 0113 R          MOV    DL,COL
09D3 B7 00                     MOV    BH,0
09D5 CD 10                     INT    10H

09D7 BF 0117 R                 MOV    DI,OFFSET BUF        ;salvar pantalla actual
09DA B9 0005                   MOV    CX,5

09DD                    RELOJ1:

09DD B4 08                     MOV    AH,8                 ;leer carácter
09DF B7 00                     MOV    BH,0
09E1 CD 10                     INT    10H
09E3 88 05                     MOV    [DI],AL
09E5 47                        INC    DI
```

EJEMPLO 6–41 (página 4 de 5)

```
09E6 FE C2              INC   DL                     ;siguiente columna
09E8 B4 02              MOV   AH,2
09EA CD 10              INT   10H
09EC E2 EF              LOOP  RELOJ1

09EE B4 02              MOV   AH,2                   ;volver a ubicar el cursor
09F0 80 EA 05           SUB   DL,5
09F3 CD 10              INT   10H

09F5 BF 011D R          MOV   DI,OFFSET BUF1

09F8 B4 2C              MOV   AH,2CH                 ;obtener la hora
09FA CD 21              INT   21H
09FC 8A C5              MOV   AL,CH
09FE 32 E4              XOR   AH,AH
0A00 D4 0A              AAM                          ;convertir décimas de hora
0A02 80 C4 20           ADD   AH,20H
0A05 80 FC 20           CMP   AH,20H
0A08 74 03              JE    RELOJ2
0A0A 80 C4 10           ADD   AH,10H

0A0D                RELOJ2:

0A0D 88 25              MOV   [DI],AH
0A0F 04 30              ADD   AL,30H
0A11 88 45 01           MOV   [DI+1],AL
0A14 2E:C6 45 02 3A     MOV   BYTE PTR CS:[DI+2],':'
0A19 8A C1              MOV   AL,CL                  ;convertir unidades de horas
0A1B 32 E4              XOR   AH,AH
0A1D D4 0A              AAM
0A1F 05 3030            ADD   AX,3030H
0A22 88 65 03           MOV   [DI+3],AH
0A25 88 45 04           MOV   [DI+4],AL
0A28 BA 011D R          MOV   DX,OFFSET BUF1
0A2B B4 09              MOV   AH,9                   ;exhibir la hora
0A2D CD 21              INT   21H
0A2F                RELOJ3:

0A2F B4 06              MOV   AH,6                   ;probar cualquier tecla
0A31 B2 FF              MOV   DL,0FFH
0A33 CD 21              INT   21H
0A35 74 F8              JZ    RELOJ3

0A37 B4 02              MOV   AH,2
0A39 2E: 8A 36 0114 R   MOV   DH,REN
0A3E 2E: 8A 16 0113 R   MOV   DL,COL
0A43 B7 00              MOV   BH,0
0A45 CD 10              INT   10H

0A47 B4 09              MOV   AH,9
0A49 BA 0017 R          MOV   DX,OFFSET BUF
0A4C CD 21              INT   21H

0A4E B4 02              MOV   AH,2
0A50 2E: 8A 36 0115 R   MOV   DH,T_REN
0A55 2E: 8A 16 0116 R   MOV   DL,T_COL
0A5A B7 00              MOV   BH,0
0A5C CD 10              INT   10H
```

EJEMPLO 6-41 (página 5 de 5)

```
0A5E 1F                     POP     DS
0A5F 5F                     POP     DI
0A60 5A                     POP     DX
0A61 59                     POP     CX
0A62 5B                     POP     BX
0A63 58                     POP     AX
0A64 C3                     RET

0A65              RELOJ     ENDP

                  ;instalar procedimientos 8 y 9 de interrupción

0A65              INICIO:

0A65 8C C8                  MOV     AX,CS
0A67 8E D8                  MOV     DS,AX

0A69 B8 3508                MOV     AX,3508H            ;obtener vector 8
0A6C CD 21                  INT     21H
0A6E 2E: 89 1E 0103 R       MOV     WORD PTR VEC8,BX
0A73 2E: 8C 06 0105 R       MOV     WORD PTR VEC8+2,ES

0A78 B8 3509                MOV     AX,3509H            ;obtener vector 9
0A7B CD 21                  INT     21H
0A7D 2E: 89 1E 0107 R       MOV     WORD PTR VEC9,BX
0A82 2E: 8C 06 0109 R       MOV     WORD PTR VEC9+2,ES
0A87 B8 2508                MOV     AX,2508H            ;instalar impulso de reloj como 8
0A8A BA 0967 R              MOV     DX,OFFSET TICK
0A8D CD 21                  INT     21H

0A8F B8 2509                MOV     AX,2509H
0A92 BA 0925 R              MOV     DX,OFFSET TECLA     ;Instalar TECLA como 9
0A95 CD 21                  INT     21H

0A97 BA 0A65 R              MOV     DX,OFFSET INICIO    ;hacer residente
0A9A C1 EA 04               SHR     DX,4
0A9D 42                     INC     DX
0A9E B8 3100                MOV     AX,3100H
0AA1 CD 21                  INT     21H

0AA3              CODE      ENDS

                            END     BEGIN
```

6-7 RESUMEN

1. El programa ensamblador ensambla módulos que contienen las variables y segmentos públicos PUBLIC y las variables EXTERNAS. El programa ligador liga los módulos y los archivos de biblioteca para producir un programa que se ejecuta desde la línea de comandos del DOS. El programa de tiempo de corrida suele tener la extensión EXE.

2. Los directivos MACRO y ENDM producen un nuevo código para empleo en programas. Estos macros son similares a los procedimientos, excepto que no hay llamada ni retorno. En lugar de ellos, el ensamblador introduce el código de la macrosecuencia en el programa cada vez que lo llama. Las macros pueden incluir variables que pasan información y datos a la macrosecuencia.
3. La llamada a funciones de la INT 21H del DOS ofrece un método para utilizar el teclado y la pantalla. La función 06H cargada en el registro AH, ofrece una interface con el teclado y la pantalla. Si DL = 0FFH, esta función prueba si se ha oprimido una tecla en el teclado; si no se detecta, retorna igual. Si detecta que se oprimió la tecla, retorna el carácter ASCII normal en AL. Si se tecleó un carácter ASCII extendido, retorna con AL = 00H y hay que volver a llamar la función para que retorne en AL el carácter ASACII extendido. Para exhibir un carácter, se carga DL con ese carácter y AH con 06H antes de utilizar la INT 21H en el programa.
4. Las cadenas de caracteres se exhiben con la función 09H. La combinación de registros DS:DX direcciona la cadena de caracteres que debe terminar con el símbolo $.
5. La instrucción INT 10H accesa a los procedimientos del BIOS que controlan el teclado y la pantalla. Las funciones del BIOS son independientes del DOS y funcionan con cualquier sistema operativo.
6. La conversión de datos de binario a BCD se logra con la instrucción AAM para números menores de 100 o con la división repetida entre 10 para números más grandes. Una vez convertidos a BCD se agrega un 30H para convertir cada dígito al código ASCII para su sexhibición en la pantalla.
7. Cuando se convierte de un número ASCI a BCD, se resta un 30H de cada dígito. Para obtener el equivalente binario, se multiplica por 10.
8. Se utilizan tablas para la conversión de código con la instrucción de XLAT si el código es de 8 bits. Si su ancho es mayor de 8 bits, entonces hay un procedimiento corto para accesar a la tabla de conversión. Las tablas también se utilizan para direcciones, a fin de poder seleccionar diferentes partes del programa o procedimientos diferentes.
9. El sistema de discos contiene pistas que llevan la información almacenada en sectores. En muchos sistemas de discos se almacenan 512 bytes de información por sector. Los datos en el disco se organizan en un sector para carga inicial, tabla de localización de archivos, directorio de raíz y zona de almacenamiento de datos. El sector de carga inicial carga el sistema DOS del disco al sistema de memoria de la computadora. La FAT indica cuáles sectores están presentes y si contienen o no datos. El directorio raíz contiene nombres de archivos y subdirectorios, por medio de los cuales se accesa a todos los archivos del disco. La zona de almacenamiento de datos contiene todos los subdirectorios y archivos de datos.
10. Los archivos se "manipulan" con las funciones de la INT 21H del DOS. Para leer un archivo de disco, hay que abrir, leer y cerrar el archivo. Para escribir en un archivo de disco, se abre, escribe y cierra. Cuando se abre un archivo el apuntador de archivo direcciona al primer byte del archivo. Para accesar a los datos en otras localidades, se mueve el apuntador antes de leer o escribir datos.
11. Un archivo de acceso secuencial es aquel al que se accesa en secuencia desde principio a fin. Un archivo de acceso aleatorio se puede accesar en cualquier punto. Aunque todos los archivos de disco son secuenciales, se pueden manejar como si fueran archivos de acceso aleatorio, con el empleo de procedimientos de programación.
12. El prefijo (PSP) de segmentos del programa contiene información acerca de un programa. Una parte importante del PSP son los parámetros de la línea de comando.

13. Los cambios de los servicios de interrupción permiten que un programa de aplicación accese o intercepte una interrupción. A menudo se emplean estos cambios en la interrupción de los impulsos del reloj (vector 8) o en el vector 9 de la interrupción del teclado.
14. Un programa de termina y permanece residente (TSR) permanece en la memoria y se accesa con frecuencia con un cambio en el servicio de la interrupción, ya sea con el impulso del temporizador o con una tecla de función especial ("caliente").
15. Una tecla de función especial ("caliente") hace funcionar un programa de termina y permanece residente por medio de un cambio del servicio de la interrupción del teclado.

6–8 CUESTIONARIO Y PROBLEMAS

1. El ensamblador convierte un archivo fuente en uno de ____________.
2. ¿Qué archivos se generan desde el archivo de fuente PRUEBA.ASM cuando se procesa con MASM?
3. El programa ligador liga los archivos objeto y los archivos de __________ para producir un archivo ejecutable.
4. ¿Qué indica el directivo PUBLIC en un módulo de programa?
5. ¿Qué indica el directivo EXTRN en un módulo de programa?
6. ¿Qué directivo aparece con las etiquetas definidas como externas?
7. Describa la forma en que trabaja un programa de biblioteca cuando está ligado con otros archivos de objeto por el programa ligador.
8. ¿Qué directivos de lenguaje ensamblador delinean una macrosecuencia?
9. ¿Qué es una macrosecuencia?
10. ¿Cómo se transfieren los parámetros a una macrosecuencia?
11. Desarrolle un macro llamado ADD32 que sume el contenido de 32 bits de DX—CX al contenido de 32 bits de BX—AX.
12. ¿Cómo se utiliza el directivo LOCAL dentro de una macrosecuencia?
13. Desarrolle un macro llamado ADDLIST PARA1,PARA2 que sume el contenido de PARA1 con el de PARA2. Cada uno de estos parámetros representa un área de la memoria. El número de bytes a sumarse los indica el registro CX antes de llamar a la macro.
14. Desarrolle un macro que sume una lista de datos de tamaño bytes llamados por el macro ADDM LISTA.LONGTD. La etiqueta LISTA es el punto de partida del bloque de datos y la longitud es el número de datos sumados. El resultado debe ser una suma de 16 bits que estará en AX al final de la macrosecuencia.
15. ¿Cuál es la finalidad del directivo INCLUDE?
16. Desarrolle un procedimiento llamado ALEATORIO. Este procedimiento debe retornar un número aleatorio de 8 bits en el registro CL al final de la subrutina.
17. Desarrolle un procedimiento que exhiba una cadena de caracteres que termina con un 00H. Se debe utilizar el registro DS:DX para direccionar el comienzo de la cadena de caracteres.
18. Desarrolle un procedimiento que lea una tecla y exhiba el valor hexadecimal de un carácter del teclado en ASCII extendido si se oprime una de tales teclas. Si se teclea un carácter normal, no lo tenga en cuenta.
19. Utilice la INT 10H del BIOS para desarrollar un procedimiento que ubique al cursor en el renglón 3, columna 6.

20. Cuando se convierte un número de binario a BCD, la instrucción _________ logra la conversión, siempre y cuando el número sea menor de 100 decimal.
21. ¿Cómo se convierte un número grande (mayor de 100 decimal) de binario a BCD?
22. Para convertir un dígito en BCD a código ASCII se le suma__________.
23. Para convertir un número en código ASCII a BCD se le resta __________.
24. Desarrolle un procedimiento para leer un número ASCII en el teclado y almacenarlo en BCD en el arreglo DATO de la memoria. El procedimiento termina cuando se teclea cualquiera que no sea un número.
25. Explique la forma en que un número de 3 dígitos en código ASCII se convierte a binario.
26. Desarrolle un procedimiento para convertir todas las letras minúsculas del código ASCII, a mayúsculas. Ese procedimiento no puede cambiar ningún carácter que no sean las letras a hasta z.
27. Desarrolle una tabla que convierta datos hexadecimales 00H a 0FH, en caracteres en ASCII que representen los dígitos hexadecimales. Muestre la tabla y cualquier programa requerido para la conversión.
28. Desarrolle una secuencia de programa que brinque a la localidad UNO en la memoria si AL=6, a la DOS si AL = 7 y a la TRES si AL = 8.
29. Muestre el empleo de la instrucción XLAT para accesar a una tabla llamada YEA, ubicada en el segmento de pila.
30. Explique la finalidad de un sector de carga inicial, de la FAT y del directorio raíz.
31. La superficie de un disco está dividida en pistas que, a su vez, están subdivididas en__________.
32. ¿Qué es un sector de inicialización y dónde se encuentra?
33. ¿Qué es un agrupamiento?
34. Una entrada al directorio contiene un byte de atributo. ¿Qué información da ese byte acerca de la entrada?
35. Una entrada al directorio contiene la longitud del disco, archivo o subdirectorio almacenados en __________ bytes de memoria.
36. ¿Cuál es la longitud máxima de un archivo?
37. Desarrolle un procedimiento que abra un archivo llamado PRBA.LST, que lea 512 bytes del archivo, los ponga en la zona ARREGLO de la memoria en el segmento de datos y cierre el archivo.
38. Desarrolle un procedimiento para cambiar el nombre del archivo PRBA.LST, a PRBA.LIS.
39. Escriba un programa que lea cualquier número decimal entre 0 y 65 535 y escriba la versión binaria de 16 bits en la pantalla.
40. Escriba un programa que muestre las potencias binarias de dos (en decimal) en la pantalla, para las potencias del 0 al 7. La pantalla muestra un valor de 2^n=, por cada potencia de dos.
41. Utilice la técnica aprendida en la pregunta 15 desarrolle un programa que exhiba números aleatorios entre el 1 y el 47 (o como sea) de la lotería básica o el "Melate" de su zona (según el caso).
42. Desarrolle un programa que exhiba el contenido hexadecimal de un bloque de 256 bytes de memoria. El programa deberá aceptar la dirección inicial como un número hexadecimal entre 00000H y FFF00H.
43. Desarrolle un programa que cambie en el vector de interrupción 0 para exhibir el siguiente mensaje al haber un error en la división: "Lo sentimos, usted ha tratado de dividir entre 0."

CAPITULO 7

Especificaciones del 8086 y 8088

INTRODUCCION

En este capítulo se describen la función de las terminales de los microprocesadores 8086 / 8088 y se dan detalles de los siguientes aspectos: generación de la señal de reloj, de multiplexión y acoplamiento del canal, temporización estados de espera y funcionamiento en modo mínimo y modo máximo.

Antes de que sea posible conectar o hacer interface de cualquier cosa en el microprocesador, es necesario entender las funciones y temporización de las terminales. Por tanto, la información de este capítulo es esencial para entender por completo la interface de memoria y de E/S memorias, que se describen en capítulos posteriores.

OBJETIVOS DEL CAPITULO

Al concluir este capítulo, usted podrá:

1. Describir la función de cada terminal del 8086/8088.
2. Entender las características de CD del microprocesador e indicar su capacidad de salida para las lógicas más comunes.
3. Utilizar el circuito integrado generador de reloj (8284A) para proporcionar la señal de reloj para el microprocesador.
4. De multiplexar y acoplar los canales del microprocesador.
5. Interpretar los diagramas de temporización.
6. Describir los estados de espera y conectar los circuitos requeridos para causar varias esperas.
7. Explicar la diferencia entre el modo de funcionamiento mínimo y el máximo.

7–1 LAS TERMINALES Y SUS FUNCIONES

En esta sección se explican la función y, en algunos casos, las funciones múltiples de cada terminal del microprocesador. Además se describen las características de CD a fin de dar una base para entender, más adelante, las secciones acerca de la de multiplexión y acoplamiento de los canales del microprocesador.

El diagrama de base

En la figura 7-1 se ilustran los diagramas de base de los microprocesadores 8086 y 8088. Al observar con cuidado, se encontrará que casi no hay diferencia entre estos dos microprocesadores: ambos están en encapsulados de 40 terminales doble en línea (DIP).

Como se mencionó en el capítulo 1, el 8086 es un microprocesador de 16 bits con un canal de datos de 16 bits y el 8088 es de 16 bits con canal de datos de 8 bits. (Como se ilustra, el 8086 tiene las conexiones de terminales AD_0 AD_{15} y el 8088 tiene las conexiones de terminales AD_0 AD_7.) Por tanto, el canal de datos es la única diferencia importante entre estos microprocesadores.

Pero, hay una pequeña diferencia en una de las señales de control. El 8086 tiene una terminal $M/\overline{IO}$ y el 8088 tiene una terminal $IO/\overline{M}$. La única otra diferencia en sus terminales aparece en la terminal 34 de ambos circuitos: en el 8088 es la terminal $\overline{SSO}$, mientras que en el 8086 es $\overline{BHE}/S$.

Requisitos del suministro de corriente

Los microprocesadores 8086 y 8088 requieren una alimentación de +5 volts con una tolerancia de ±10%. El 8086 tiene un consumo máximo de corriente de 360 mA y el 8088, de 340 mA. Ambos microprocesadores funcionan a temperaturas ambientes entre 0°C (32°F) y alrededor de 82°C

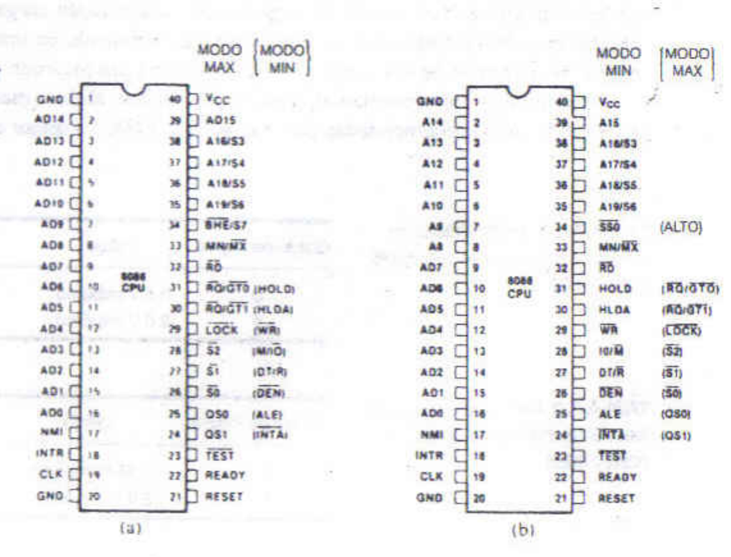


FIGURA 7–1 (a) conexiones de terminales en el microprocesador 8086;(b) conexiones de terminales del microprocesador 8088.

(180°F). Este intervalo no es lo bastante amplio para emplearlos en climas cálidos o fríos extremosos, pero se dispone de versiones para mayor capacidad de temperatura de los microprocesadores 8086 y 8088. También hay una versión CMOS, que requiere un suministro de corriente muy bajo y tiene un amplio intervalo de temperatura. Los 80C88 y 80C86 son versiones de CMOS que requieren sólo 10 mA de corriente y funcionan en temperaturas extremosas entre −50°C (−40°F) y 108°C (225°F).

Características de la CD

Es imposible conectar algo con las terminales del microprocesador sin saber cuál es la corriente de entrada requerida en una terminal de entrada y cuál es la capacidad de corriente de salida en una terminal de salida. Este conocimiento permite al diseñador seleccionar los componentes de interface adecuados para el microprocesador sin miedo de dañar nada.

Características de la entrada. Las características de entrada de estos microprocesadores son compatibles con todos los componentes estándar lógicos disponibles. En la tabla 7-1 se presentan los valores de voltaje de entrada y también los de corriente de entrada requerida para cualquier terminal en cualquiera de los microprocesadores. Los valores de corriente de entrada son muy bajos porque las entradas son las conexiones a compuertas MOSFET y representan sólo corrientes de fuga.

Características de la salida. En la tabla 7-2 se muestran las características de salida de todas las terminales de salida de los microprocesadores. El valor de voltaje del 1 lógico de los 8086/8088 es compatible con el de la mayor parte de los componentes lógicos estándar, pero el valor 0 lógico no lo es. Los circuitos (estándar) lógicos tienen un voltaje 0 lógico máximo de 0.4 volt y en los 8086/ 8088 es un máximo de 0.45 volt. Por tanto, hay una diferencia de 0.05 volt.

Esta diferencia reduce la inmunidad al ruido desde un valor normal de 400 mV (0.8 V a 0.45 V) a 350 mV. (La inmunidad al ruido es la diferencia entre los valores de voltaje del 0 lógico de salida y del 0 lógico de entrada.) Esta inmunidad reducida contra el ruido puede ocasionar problemas en conexiones con alambres largos o con demasiadas cargas. Por ello, se recomienda no conectar más de 10 cargas de cualquier tipo o combinación en una terminal de salida sin acoplamiento. Si se excede de esa carga, el ruido aparecerá produciendo problemas en la temporización.

En la tabla 7-3 se presentan algunas de las familias lógicas más comunes y las capacidades de corriente de salida recomendadas para los 8086 y 8088. La mejor elección de los tipos de compo-

TABLA 7–1 Características de entrada a los microprocesadores 8086 y 8088

Grado de lógica	*Voltaje*	*C. Corriente*
0	0.8 V máximo	10 microamp. max.
1	2.0 V máximo	10 microamp. max.

TABLA 7–2 Características de salida de los microprocesadores 8086 y 8088

Grado de lógica	*Voltaje*	*Corriente*
0	0.45 V máximo	2.0 microamp. máximo.
1	2.4 V máximo	-400 microamp. máximo

TABLA 7-3 Divergencia de salida recomendada para cualquier conexión de terminal en 8086 y 8088

Familia	*Divergencia*	*Corriente de caída*	*Corriente de fuente*
TTL (74XX)	1	−1.6 mA	40 μA
TTL(74LSXX)	5	−0.4 mA	20 μA
TTL (74ALSXX)	1	−2.0 mA	50 μA
TTL (74ALSXX)	10	−0.2 mA	20 μA
CMOS (74HCXX)	10	−1.0 μA	1.0 μA
CMOS (CD4XXX)	10	−1.0 μA	1.0 μA
NMOS	10	−10 μA	10 μA

nentes para conexión con una terminal de salida del 8086 u 8088 es un componente lógico LS,74ALS o 74HC.

Terminales de conexión

1. AD_7–AD_0 (8088). *Canal de direcciones y de datos*: constituyen el canal multiplexado de direcciones y de datos del 8088 y contiene los 8 bits más a la derecha de la dirección de memoria o del número de puerto de E/S, siempre que ALE está activo (1 lógico) o de los datos, siempre que ALE no está activo (0 lógico). Estas terminales se encuentran en su estado de alta impedancia durante un reconocimiento de solicitud del canal.
2. A_{15}–A_8 (8088). *Canal de direcciones*: constituyen la mitad superior de los de la dirección de memoria que se encuentran presentes durante un ciclo del canal. Estas terminales para la dirección su estado de alta impedancia durante un reconocimiento de solicitud de canal.
3. AD_{15}–AD_7 (8086). *Canal de direcciones y de datos*: las líneas que constituyen la parte superior del canal multiplexado de direcciones y de datos en el 8086. Estas líneas contienen los bits de dirección A_{15} A_8 siempre que ALE es un 1 lógico y a D_{15} D_8 del canal de datos. Cuando ALE es un cero lógico estas terminales pasan a un estado de alta impedancia siempre que ocurre un reconocimiento de solicitud de canal.
4. A_{19} y S_6, A_{18} y S_5, A_{17} y S_4 y A_{16} y S_3. *Canal de direcciones y estado*: están multiplexadas para suministrar señales A_{19} A_{16} de dirección y, también los bits S_6 a S_3 de estado. Estas terminales también alcanzan un estado de alta impedancia durante el reconocimiento de una solicitud de canal.

   El bit de estado S_6 siempre es un 0 lógico; el bit S_5 indica la condición del bit de bandera IF y los bits S_4 y S_3 muestran a cuál segmento se está accesando durante el ciclo de canal actual. En la tabla 7-4 aparece una tabla de verdad para S_4 y S_3. Estos dos bits de estado se podrían utilizar para direccionar cuatro bancos separados de memoria de 1 M byte.
5. $\overline{RD}$. *Lectura*: señala (cuando es 0 lógico) que el canal de datos puede recibir datos de la memoria o de los dispositivos de E/S conectados en el sistema. Esta terminal "flota" a su estado de alta impedancia durante un reconocimiento de una solicitud de canal.
6. READY. *Lista*: es una entrada que cuando se controla se puede emplear para introducir estados de espera en la temporización del microprocesador. Si esta terminal se lleva a un 0 lógico, el microprocesador introduce estados de espera y permanece en el mismo ciclo del canal. Si se lleva a esta terminal a 1 lógico, no tiene efecto en el funcionamiento del microprocesador.

TABLA 7–4 Función de "S_4 AND S_3"

S_4	S_3	*Función*
0	0	Segmento adicional
0	1	Segmento de pila
1	0	Código o no hay segmento
1	1	Segmento de datos

7. INTR. *Solicitud de interrupción*: Se emplea para solicitar una interrupción por hardware. Si INTR se mantiene alta cuando IF = 1, los microprocesadores entran en un ciclo de reconocimiento de interrupción, $\overline{\text{INTA}}$ se activa una vez que se ha ejecutado la instrucción presente.
8. $\overline{\text{TEST}}$. *Prueba*: es una terminal de entrada que prueba la instrucción WAIT. Si $\overline{\text{TEST}}$ es un 0 lógico, la instrucción WAIT funciona como una No operación. Si $\overline{\text{TEST}}$ es un 1 lógico, entonces la instrucción WAIT espera a que $\overline{\text{TEST}}$ sea un 0 lógico. Muchas veces esta terminal está conectada frecuentemente en el coprocesador numérico 8087.
9. NMI. *Interrupción no enmascarable*: es similar a INTR excepto que NMI no verifica si el bit de bandera IF es un 1 lógico. Si se hace funcionar NMI, esta entrada de interrupción emplea el vector de interrupción 2.
10. RESET. *Inicializar*: hace que el microprocesdor se inicialice por sí solo, si esta terminal se mantiene en un nivel alto (positiva) durante un mínimo de 4 periodos de reloj. Cuando se reinicializan el 8086 o el 8088, empiezan a ejecutar instrucciones en la localidad FFFF0H de la memoria y deshabilita las futuras interrupciones desactivando el bit del registro de banderas IF.
11. CLK. *Reloj*: produce la señal básica de temporización para el microprocesador. La señal de reloj debe tener un ciclo de trabajo de 33% (alto durante un tercio del periodo de reloj y bajo los dos tercios restantes) a fin de producir la temporización interna adecuada para el 8086 y 8088.
12. V_{cc}. *Alimentación*: la entrada V_{cc} produce una señal de 5.0 volts ±10% al microprocesador.
13. GND, *Tierra*: es la conexión de retorno del suministro de corriente. Se debe tener en cuenta que en los 8086 y 8088 hay dos terminales GND y ambas deben estar conectadas a tierra, para tener buen funcionamiento.
14. MN/$\overline{\text{MX}}$. *Modo mínimo o máximo*: selecciona el modo de funcionamiento mínimo o máximo para el microprocesador. Si se selecciona el modo mínimo, la terminal MN/$\overline{\text{MX}}$ debe estar conectada en forma directa con +5.0 volts.
15. $\overline{\text{BHE}}$/S_7. *Habilitación de la parte alta del canal*: se utiliza en el 8086 para habilitar los bits más significativos del canal de datos (D_{15} a D_8) durante una operación de lectura o escritura. El estado de S_7 es siempre un 1 lógico.

Terminales en modo mínimo. El funcionamiento en modo mínimo de los 8086 y 8088, se obtiene al conectar la terminal MN/$\overline{\text{MX}}$ directamente con 5.0 volts. No conecte esta terminal a 5.0 volts a través de un resistor para nivel alto o no funcionará bien.

1. IO/$\overline{\text{M}}$ (8088), M/$\overline{\text{IO}}$ (8086). *Memoria o entrada/salida*: es una terminal que indica a la memoria y a E/S que el canal de direcciones del microprocesador contiene, ya sea una dirección de

memoria o bien una dirección de puerto de E/S. Esta terminal está en alta impedancia durante un reconocimiento de solicitud de canal.

2. $\overline{WR}$. *Escribir*: es una señal de habilitación que indica que los 8086 y 8088 están sacando datos a memoria o a un dispositivo de E/S. Durante el tiempo en que $\overline{WR}$ es un 0 lógico, el canal de datos contiene datos válidos para la memoria o E/S. Esta terminal flota a una alta impedancia durante un reconocimiento de solicitud del canal.
3. $\overline{INTA}$. *Reconocimiento de interrupción*: es una respuesta a la terminal de entrada INTR. La terminal $\overline{INTA}$ se suele utilizar para presentar el número del vector de interrupción al canal de datos en respuesta a una solicitud de interrupción.
4. ALE. *Habilitación de la dirección*: muestra que el canal de direcciones/datos del 8086 y el 8088 contiene una dirección. Esta dirección puede ser de memoria o un número de puerto de E/S. Se debe tener en cuenta que la señal ALE no flota durante un reconocimiento de la solicitud de canal.
5. DT/$\overline{R}$. *Transmitir y recibir datos:* muestra que el canal de datos transmite datos (DT/$\overline{R}$ =1) o los recibe (DT/$\overline{R}$ = 0). Esta señal se utiliza para habilitar los circuitos de acoplamiento externos al canal de datos.
6. $\overline{DEN}$. *Habilitar canal de datos*: se emplea para habilitar los circuitos de acoplamiento del canal externo a datos.
7. HOLD. *Retener*: solicita un acceso directo a la memoria (DMA). Si la señal HOLD es un 1 lógico, el microprocesador deja de ejecutar el programa actual y lleva su canal de direcciones, datos y control al estado de alta impedancia. Si la terminal HOLD es un 0 lógico, el microprocesador ejecuta normalmente los programas.
8. HLDA. *Reconocimiento*: indica que los 8086 u 8088 han concedido los canales, por lo regular, para un ciclo de DMA.
9. $\overline{SSO}$. (8088): *Línea de estado* 0: equivale a la terminal $\overline{SO}$ en el modo de funcionamiento máximo del microprocesador. Esta señal se combina con IO/$\overline{M}$ y con DT/$\overline{R}$ para decodificar la función del ciclo de canal actual (véase tabla 7-5).

Terminales para modo máximo. A fin de obtener modo máximo para empleo con coprocesadores externos, se conecta la terminal MN/$\overline{MX}$ a tierra.

1. $\overline{S2}$, $\overline{S1}$ y $\overline{S0}$. *Bits de estado*: indican la función del ciclo del canal actual. Estas señales se suelen decodificar con el controlador de canal 8288 que se describe más adelante en este capítulo. En la tabla 7-6 se muestra la función de estos tres bits de estado en el modo máximo.

TABLA 7-5 Estado del ciclo de canal con el empleo de $\overline{SSO}$ en el 8088

IO/$\overline{M}$	*DT/$\overline{R}$*	*$\overline{SSO}$*	*Función*
0	0	0	Reconocer interrupción
0	0	1	Leer memoria
0	1	0	Escribir en memoria
0	1	1	Detener
1	0	0	Acceder a código
1	0	1	Leer E/S
1	1	0	Escribir E/S
1	1	1	Pasiva

TABLA 7-6 Funciones de control del canal generadas por el control ce canal (8288) con el empleo de $\overline{S2}$, $\overline{S1}$ y $\overline{SO}$

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	*Función*
0	0	0	Reconocer interrupción
0	0	1	Leer E/S
0	1	0	Escribir E/S
0	1	1	Detener
1	0	0	Acceder a código
1	0	1	Leer memoria
1	1	0	Escribir en memoria
1	1	1	Pasiva

TABLA 7-7 Bits de estado de la cola

QS1	*QS0*	*Función*
0	0	No hay operación (cola está ociosa)
0	1	Primer byte de un opcode
1	0	La cola está vacía
1	1	Byte subsecuente de un opcode

2. $\overline{RQ/GT0}$ y $\overline{RQ/GT1}$. *Terminales de solicitud y otorgamiento*: terminales que solicitan acceso directo a memoria (DMA) durante el funcionamiento en modo máximo. Cada una de estas líneas es bidireccional y se emplea para solicitar y para otorgar una operación de DMA.
3. $\overline{LOCK}$. *Bloquear*: es una salida que se emplea para bloquear periféricos del sistema. Para activar esta terminal, se emplea el prefijo LOCK en cualquier instrucción.
4. QS1 y QS0. *Estados de cola de espera*: muestran el estado de la cola interna de instrucciones. Estas terminales se proporcionan para que el coprocesador numérico (8087) tenga acceso. En la tabla 7-7 se presenta el funcionamiento de los bits de estado de la cola.

7–2 GENERADOR DE RELOJ (8284A)

En esta sección se describen el generador de reloj (8284A), la señal de reinicialización RESET y la señal Lista READY para los microprocesadores 8086/8088. La señal LISTA y los circuitos relativos se describen en detalle en la sección 7-5.

El generador de reloj 8284A

El generador de reloj 8284A es un componente adicional para los microprocesadores 8086 y 8088. Si no hubiera un generador de reloj, se necesitarían muchos circuitos adicionales para generar la señal de reloj (CLK) en un sistema basado en 8086/8088. El generador 8284A produce las siguientes funciones o señales básicas: generación de reloj, sincronización de reinicialización,

sincronización de la señal LISTA y señal periférica de reloj de nivel TTL. En la figura 7-2 se ilustra el diagrama de base de un generador de reloj 8284A.

Funciones de las terminales. El 8284 es un circuito integrado con 18 terminales destinado en forma específica para emplearlo en los microprocesadores 8086/8088. A continuación aparece la lista de las terminales y su función.

1. $\overline{\text{AEN1}}$ y $\overline{\text{AEN2}}$. *De habilitación de dirección*: se proporcionan terminales para calificar a las señales de dirección RDY1 y RDY2 del canal. En la sección 7-5 se ilustra el empleo de estas dos terminales, que se emplean para ocasionar estados de espera, junto con las entradas RDY1 y RDY2. Los estados de espera se generan en la terminal READY de los microprocesadores 8086/8088, la cual se controla con estas dos entradas.
2. RDY1 y RDY2. *Canal listo*: las entradas se producen en conjunción con las terminales $\overline{\text{AEN1}}$ y $\overline{\text{AEN2}}$ para producir estados de espera en el sistema basado en el 8086/8088.
3. $\overline{\text{ASYNC}}$. *Seleccionar sincronización de lista:* es una entrada que se emplea para seleccionar una o dos etapas de sincronización para las entradas RDY1 y RDY2.
4. READY. *Lista:* esta terminal de salida se conecta con la entrada READY de los 8086/8088. Se sincroniza con las entradas RDY1 y RDY2.
5. X1 y X2. *Entradas de cristal*: estas terminales se conectan con un cristal externo utilizado como la fuente de temporización para el generador de reloj y todas sus funciones.
6. F/$\overline{\text{C}}$. *Frecuencia cristal*: selecciona la fuente de reloj para el 8284A. Si esta terminal se mantiene alta, se aplica un reloj externo a la terminal de entrada EFI y si se mantiene baja, el oscilador interno a cristal produce la señal de temporización.
7. EFI. *Entrada de frecuencia externa*: una entrada que se emplea cuando se lleva a nivel alto la terminal F/$\overline{\text{C}}$. EFI suministra la temporización siempre que la terminal F/$\overline{\text{C}}$ está alta.
8. CLK. *Reloj*: una terminal de salida en la que se proporciona la señal CLK a los microprocesadores 8086 y 8088 y otros componentes del sistema. La terminal CLK tiene una señal de salida que es un tercio de la frecuencia del cristal o de EFI y tiene un ciclo de trabajo de 33%, que es la que requieren los 8086 y 8088.
9. PCLK. *Reloj periférico*: una señal que es 1/6 parte de la frecuencia del cristal o de entrada EFI y tiene un ciclo de trabajo del 50%. La salida PCLK suministra una señal de reloj al equipo periférico del sistema.
10. OSC. *Salida de oscilador*: una señal de nivel TTL que está a la misma frecuencia que la entrada de cristal o EFI. La salida OSC proporciona una entrada EFI a otros generadores de reloj 8284A en algunos sistemas multiprocesadores.

FIGURA 7-2 Conexiones de terminales en el generador de reloj 8284A.

11. $\overline{RES}$. *Entrada de reinicialización*: una entrada activa en bajo para reinicializar al 8284A. La terminal $\overline{RES}$ a menudo está conectada a una red de RC que proporciona reinicialización al encendido.
12. RESET. *Salida de reinicialización*: la señal conectada a la terminal RESET del 8086/8088.
13. CSYNC. *Sincronización de reloj*: terminal que se emplea siempre que la entrada EFI proporciona la temporización en sistemas multiprocesadores. Si se utiliza el oscilador interno a cristal, esta terminal debe estar conectada a tierra.
14. GND. *Tierra*: una terminal de conexión a tierra.
15. V_{cc}. *Entrada de voltaje de alimentación:* terminal que se conecta a +5.0 volts con una tolerancia de ±10%.

Funcionamiento del 8284A

Es bastante fácil entender el funcionamiento del 8284A. En la figura 7-3 se presenta el diagrama lógico interno del generador de reloj 8284A.

Funcionamiento de la sección de reloj. La mitad superior del diagrama lógico representa la sección de reloj y de sincronización de reinicialización del 8284A. Como se ve en el diagrama, el oscilador de cristal tiene dos entradas: X1 y X2. Si se agrega un cristal a X1 y X2, el oscilador genera una señal de onda cuadrada a la misma frecuencia que la del cristal. La señal de onda cuadrada se alimenta a una compuerta AND y también a un inversor acoplador que produce la señal de salida OSC; ésta, a veces, se emplea como entrada EFI para otros 8284A.

El examen de la compuerta AND revela que cuando F/$\overline{C}$ es un 0 lógico la salida del oscilador se dirige hasta el contador de dividir entre 3. Si F/$\overline{C}$ es un 1 lógico, entonces se envía EFI hacia el contador.

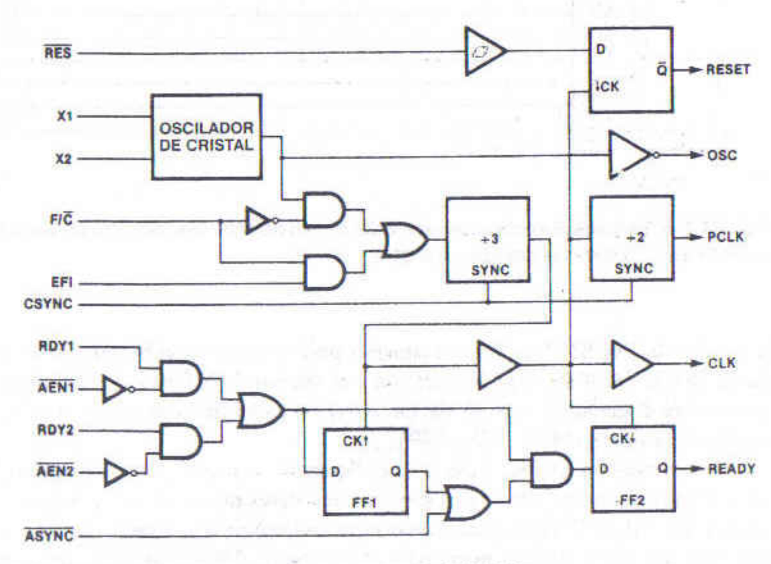


FIGURA 7-3 Diagrama de bloque interno del generador de reloj 8284A.

La salida del contador divisor entre 3 genera la temporización para la señal RDY; una señal a otro contador (divisor entre 2) y la señal CLK para los microprocesadores 8086/8088. La señal CLK también se acopla para aumentar su capacidad de corriente antes que salga del generador de reloj. Se debe tener en cuenta que la salida del primer contador alimenta al segundo. Estos dos contadores en cascada producen la salida dividida entre 6 en PCLK, la salida de reloj periférico.

En la figura 7-4 se ilustra la forma de conectar el generador 8284A con los microprocesadores 8086/8088. Se debe tener en cuenta (1) que F/$\overline{C}$ y CSYNC se conectan a tierra para seleccionar el oscilador de cristal y (2) que un cristal de 15 MHz produce la señal de reloj normal de 5 MHz para los microprocesadores, así como una señal de reloj periférico de 2.5 Mhz.

Funcionamiento de la sección de reinicialización. La sección de reinicialización del 8284A es muy sencilla. Consiste de un disparador Schmitt y un solo circuito flip-flop del tipo D; éste garantiza que se cumpla con los requisitos de temporización de la entrada RESET de los 8086/8088. Este circuito aplica la señal RESET en el flanco negativo (transición de 1 a 0) de cada ciclo de reloj. Los microprocesadores 8086 y 8088 muestrean RESET en el positivo (transición 0 a 1) de los ciclos relojes; por tanto, este circuito cumple con los requisitos de temporización de 8086 y 8088.

Con referencia de nuevo a la figura 7-4, se verá que un circuito RC aplica un 0 lógico a la entrada $\overline{\text{RES}}$ siempre que se enciende el sistema. Después de un corto tiempo, la entrada $\overline{\text{RES}}$/ se vuelve un 1 lógico porque el capacitor se carga hacia +5.0 volts a través del resistor. Un interruptor de botón permite al operador inicializar al microprocesador. La temporizacióin correcta de la reinicialización requiere que la entrada RESET se vuelva un 1 lógico, a más tardar cuatro impulsos de reloj después que se enciende el sistema, y que se mantenga alta, cuando menos 50 microsegundos. El flip-flop determina que RESET se vuelva alta en cuatro pulsos de reloj y la constante de tiempo RC asegura que permanecerá alta, cuando menos, 50 microsegundos.

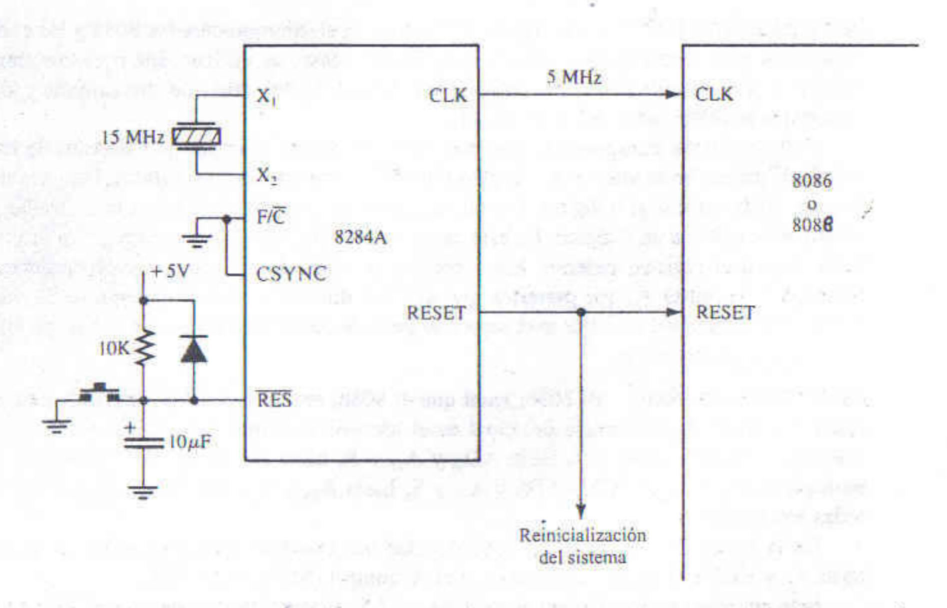


FIGURA 7-4 El generador de reloj 8284A y los microprocesadores 8086 y 8088 para ilustrar la conexión para las señales de reloj y de reinicialización. Un cristal de 15 MHz produce el reloj de 5 MHz en el microprocesador.

7–3 DEMULTIPLEXION Y ACOPLAMIENTO DEL CANAL

Antes de poder utilizar los 8086/8088 con memoria o interfaces de E/S, hay que demultiplexar los canales que están multiplexados. En esta sección se da la información detallada requerida para demultiplexar los canales y para describir la forma en que los canales se acoplan para sistemas muy grandes. (Debido a que la capacidad de la corriente de salida es de 10, hay que acoplar si el sistema contiene más de 10 componentes.)

Demultiplexión de los canales

El canal de dirección y de datos de los 8086/8088 está multiplexado para reducir el número de terminales requeridas en el circuito integrado de estos microprocesadores. Desafortunadamente, esto aumenta el trabajo del diseñador con la tarea de extraer o demultiplexar la información contenida en estas terminales multiplexadas.

¿Por qué no dejar multiplexados los canales? La memoria y la E/S requieren que la dirección siga siendo válida y estable en un ciclo de lectura o escritura. Si los canales están multiplexados, hay cambios de dirección en la memoria y en la E/S lo que hace leer o escribir datos en localidades erróneas.

Todos los sistemas de computadoras tienen tres canales: (1) un canal de direcciones que proporciona dirección de memoria al número de puerto para la E/S; (2) un canal de datos que transfiere los datos entre el microprocesador y la memoria y la E/S en el sistema; (3) un canal de control que aplica señales de control en la memoria y en E/S. Se necesitan estos tres canales para la interface con la memoria y la E/S.

Demultiplexión del 8088. En la figura 7-5 se ilustran el microprocesador 8088 y los componentes requeridos para demultiplexar sus canales. En este caso, se utilizan dos registros transparentes 74LS373 para demultiplexar las conexiones AD_7–AD_0 del canal de direcciones y datos y las terminales multiplexadas A_{19} y S_6–A_{16}/S_3.

Estos registros transparentes, que son como alambres, siempre que la señal de habilitación (ALE) de dirección se vuelve un 1 lógico transfiere las entradas a las salidas. Después de un corto tiempo, ALE regresa al 0 lógico, lo cual hace que los registros recuerden las entradas en el momento del cambio a un 0 lógico. En este caso, "recuerdan" de A_7–A_0 en el registro inferior y a A_{19} hasta A_{16} en el registro superior. Esto produce un canal de direcciones independientes para las terminales A_{19} hasta A_0 que permiten que el 8088 direccione 1 MB de espacio de memoria. El hecho de que el canal de datos esté separado permite conectarlo con cualquier dispositivo periférico de 8 bits o memoria.

Demultiplexión del 8086. El 8086, igual que el 8088, requiere canales separados de direcciones, datos y control. La diferencia principal es el número de terminales multiplexadas. En el 8088, sólo están multiplexadas AD_7 hasta AD_0, y A_{19} y S_6 hasta A_{16} y S_3. En el 8086, las terminales multiplexadas incluyen AD_{15}–AD_0, y A_{19} y S_6 hasta A_{16} y S_3 y $\overline{BHE}/S_7$. Hay que demultiplexar todas estas señales.

En la figura 7-6 se ilustra un 8086 con los tres canales de multiplexados: (direcciones A_{19} hasta A_0 y $\overline{BHE}$); el de datos (D_{15} a D_0 y el de control ($M/\overline{IO}$, RD Y $\overline{WR}$).

Este circuito es casi idéntico al de la figura 7-5, excepto que se agregó un 74SL373 adicional para demultiplexar las terminales direcciones/datos del canal AD_{15}-AD_8 y $\overline{BHE}/SE_7$ se agregó a

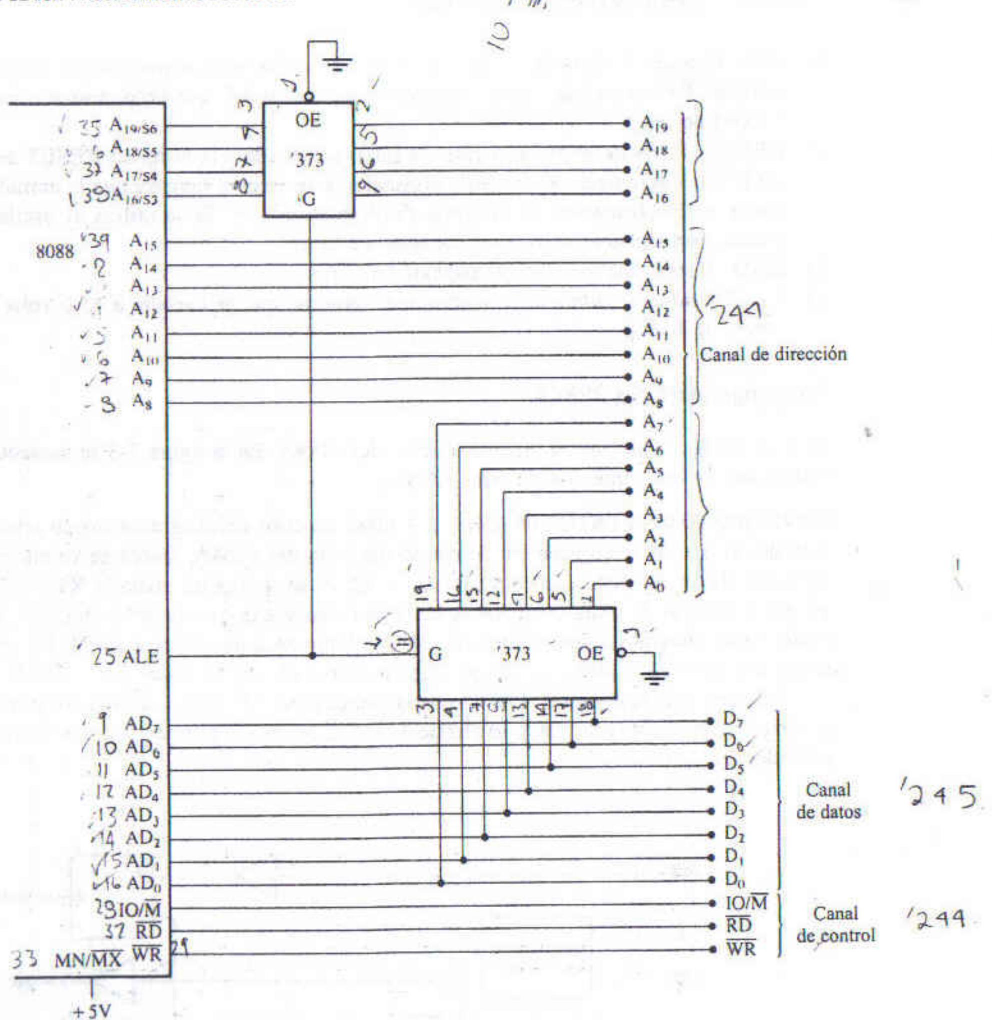


FIGURA 7–5 Se ilustra el microprocesador 8088 con un canal de dirección demultiplexado. Es el modelo para construir muchos sistemas basados en el 8088.

la entrada del 74LS373 de la parte superior para seleccionar el banco alto en el sistema de memoria de 16 bits del 8086. Aquí la memoria y el sistema de E/S ven al 8086 como un dispositivo con un canal de direcciones o de 20 bits (A_{19}–A_0), un canal de datos de 16 bits (D_{15}–D_0), y un canal de control de 3 líneas ($M/\overline{IO}$, $\overline{RD}$ y $\overline{WR}$).

Este circuito es casi idéntico al de la figura 7-5, excepto que se agregó un 74LS373 adicional para demultiplexar las terminales direcciones / datos del canal AD_{15}-AD_8 y $\overline{BHE}/S_7$ se agregó la entrada del 74LS373 de la parte superior para seleccionar el banco alto en el sistema de memoria de 16 bits del 8086. Aquí la memoria y el sistema de E/S ven al 8086 como un dispositivo con un

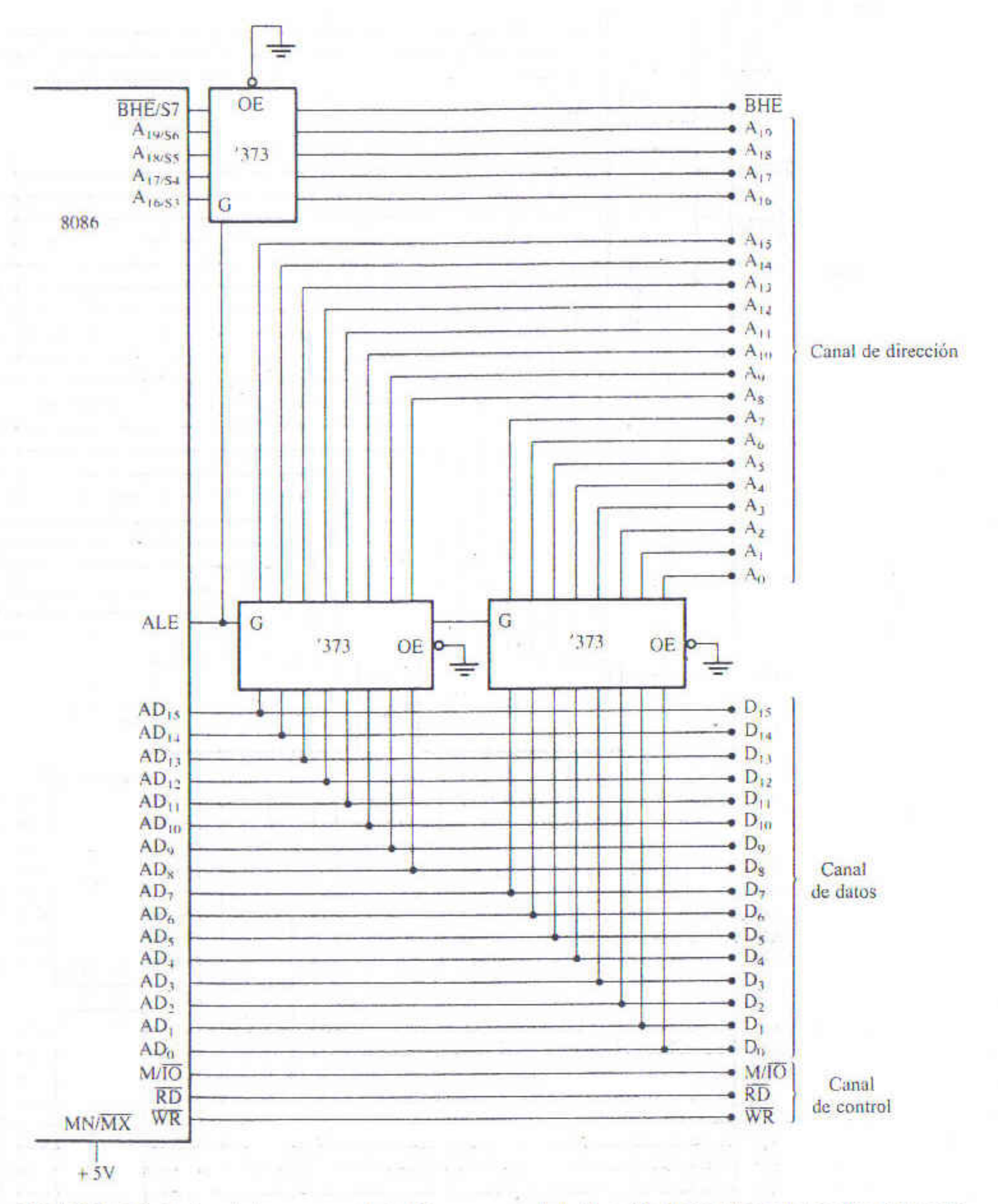


FIGURA 7–6 Se ilustra el microprocesador 8086 con un canal de dirección demultiplexado. Es el modelo para construir muchos sistemas basados en el 8086.

canal de direcciones o de 20 bits (A_{19}–A_0), un canal de datos de 16 bits (D_{13}–D_0), y un canal de control de 3 líneas ($M/\overline{IO}$, $\overline{RD}$, y $\overline{WR}$).

El sistema acoplado

Si hay más de 10 cargas unitarias conectadas con cualquier terminal de un canal, hay que acoplar la totalidad del sistema del 8086 o del 8088. Las terminales demultiplexadas ya están acopladas con los registros transparentes 74LS373, que se han diseñado para manejar los canales de alta capacitancia que se encuentran en los sistemas de microcomputadoras. Se han aumentado las corrientes de salida de los registros a fin de que se puedan manejar más unidades de cargas carga TTL; un 0 lógico de salida proporciona hasta 32 mA de disipación de corriente y una salida en 1 lógico proporciona hasta 5.2 mA de corriente.

Una señal acoplada introducirá un retardo al sistema. Esto no produce ninguna dificultad, salvo que se utilicen memoria o dispositivos de E/S, que funcionan a la máxima velocidad del canal o cerca de ella. En la sección 7-4 se comentan con mayor detalle este problema y los retardos de tiempo ocasionados.

El 8088 totalmente acoplado. En la figura 7-7 se ilustra un microprocesador 8088 totalmente acoplado. Se verá que las ocho terminales restantes A_{15}–A_8 de dirección emplean un registro transparente octal 74LS244, en las ocho terminales del canal de datos, D_7–D_0 se emplea un registro octal bidireccional 74LS245; las señales del canal de control, $IO/\overline{M}$, $\overline{RD}$ y $\overline{WR}$ se utiliza un registro 74LS244. Un sistema totalmente acoplado para el 8088 requiere un 74LS244, un 74LS245 y dos 74LS373. La dirección del 74LS245 se controla con la señal $DT/\overline{R}$ y se habilita y deshabilita con la señal $\overline{DEN}$.

El 8086 totalmente acoplado. En la figura 7-8 se ilustra un microprocesador totalmente acoplado o multiplexado. Sus terminales de dirección ya están acopladas por registros transparentes 74LS373; en el canal de datos se emplean dos registros octales bidireccionales 74LS245; para las señales del canal de control, $M/\overline{IO}$, $\overline{RD}$ y $\overline{WR}$ se utiliza un registro 74LS244. Un sistema 8086 totalmente acoplado requiere un 74LS244, dos 74LS245 y tres 74LS373. En el 8086 se requiere un acoplador más que en el 8088 debido a las ocho conexiones adicionales D_{15}–D_8 del canal de datos. También tiene la señal $\overline{BHE}$ que se acopla para seleccionar en el banco de memoria.

7–4 TEMPORIZACION DEL CANAL

Es esencial entender la temporización del canal del sistema antes de seleccionar un dispositivo de memoria o de E/S para conectarlo con un microprocesador 8086/8088. En esta sección se describen el funcionamiento de las señales del canal y la temporización básica de lectura y escritura en el 8086/8088. Es importante tener en cuenta que en esta sección sólo se describen los tiempos que influyen en la interconexión o interface de memoria y de E/S.

Funcionamiento básico de los canales

Los tres canales —dirección, datos y control— del 8086 y el 8088 funcionan exactamente en la misma forma que los de cualquier otro microprocesador. Si se escriben datos en la memoria (véase la sincronización simplificada para escritura en la figura 7-9), el microprocesador da sali-

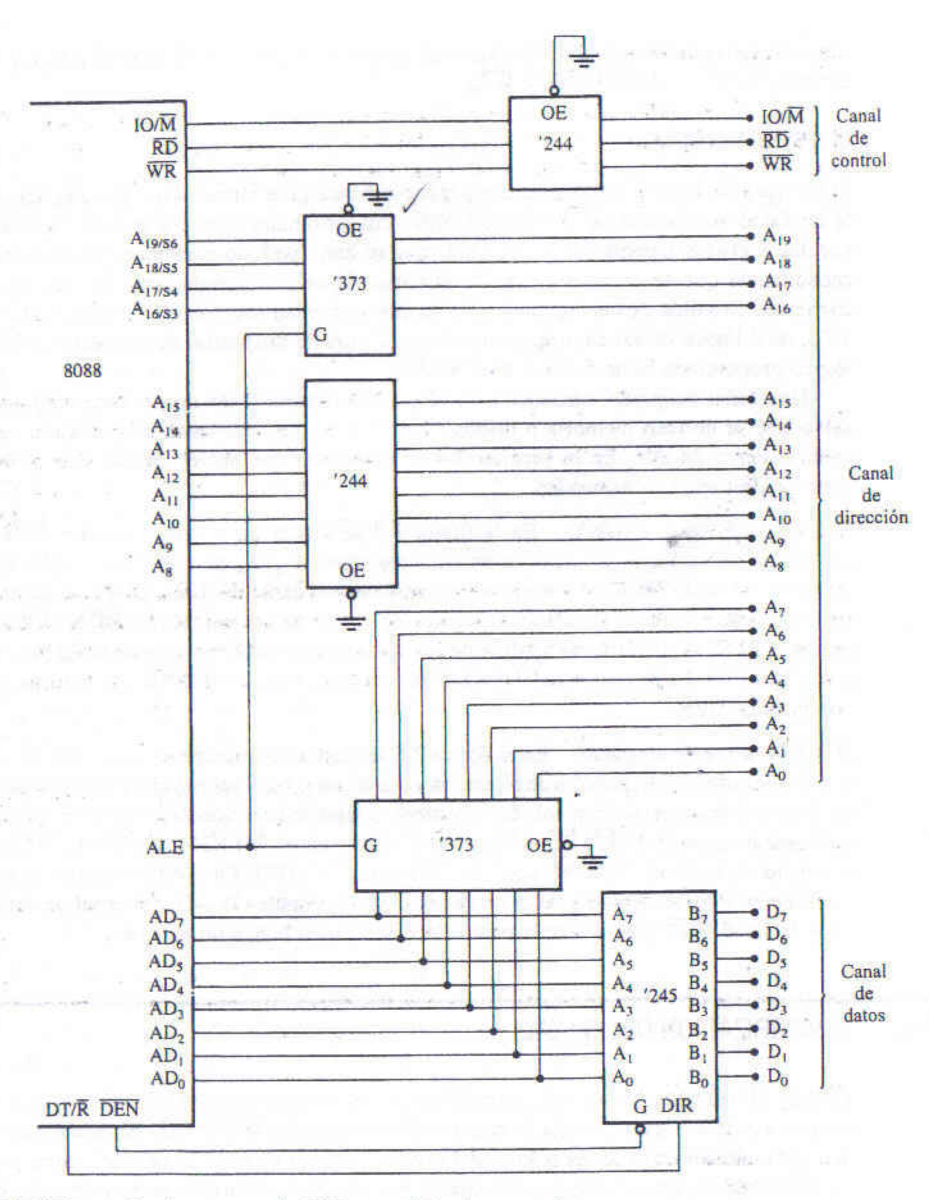


FIGURA 7–7 Un microprocesador 8088 con registros transparentes.

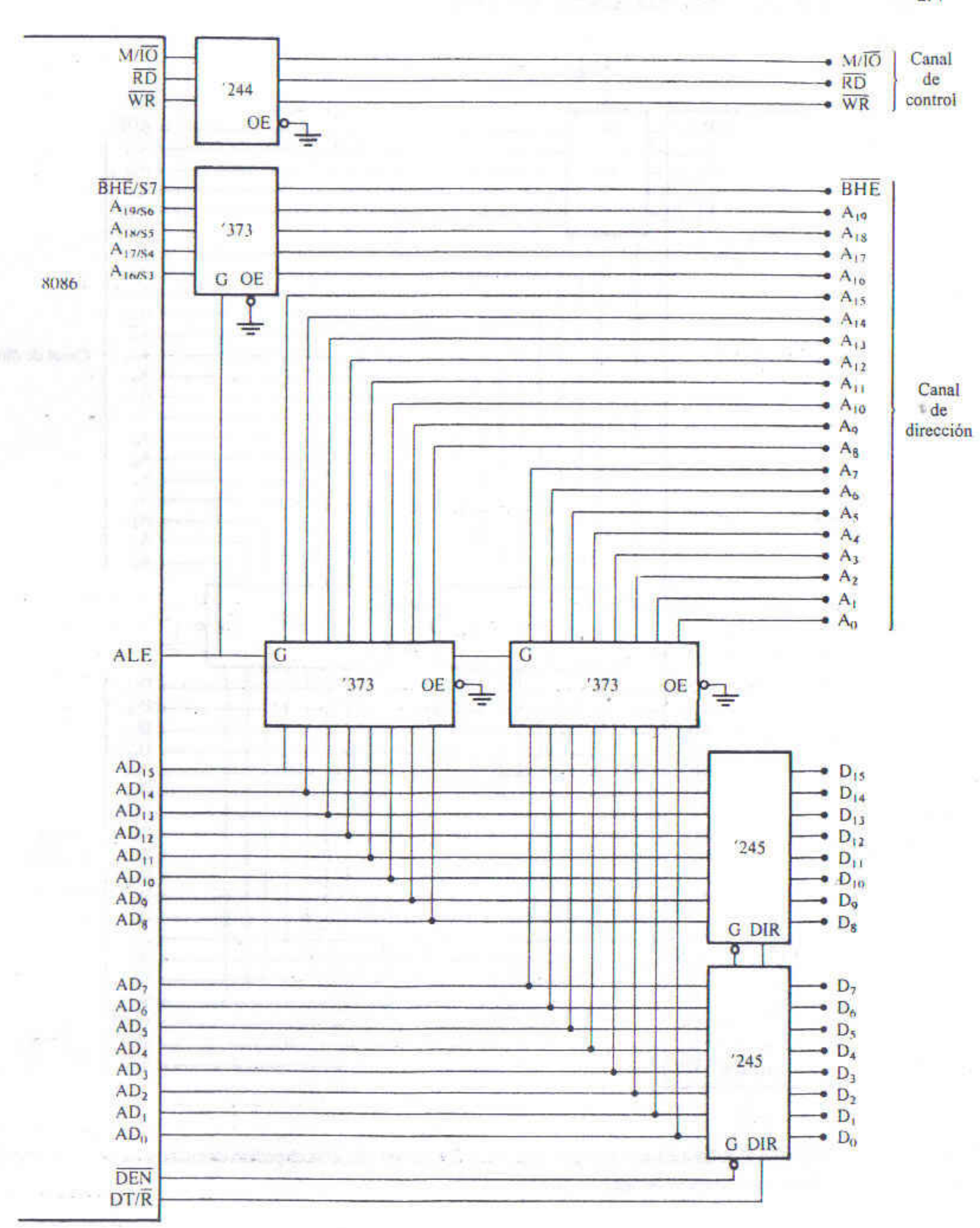


FIGURA 7–8 Un microprocesador 8086 con registros transparentes.

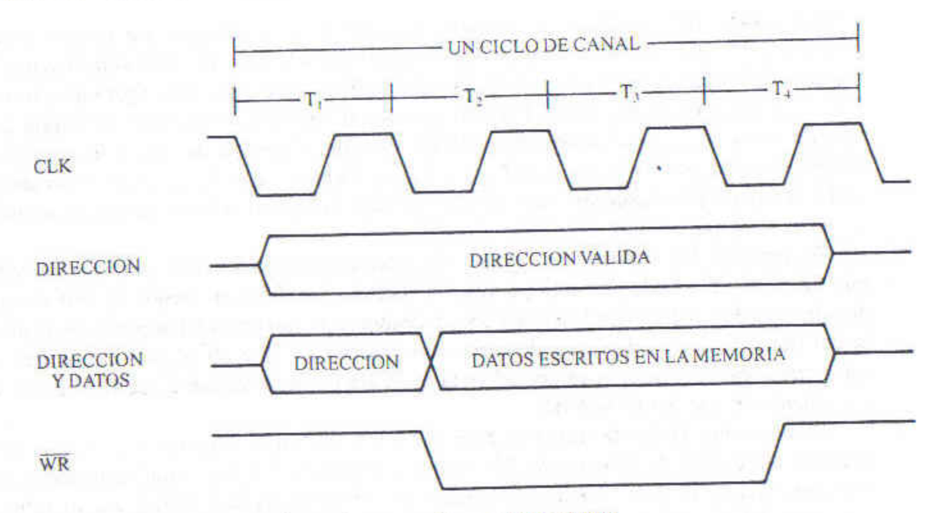


FIGURA 7–9 Ciclo simplificado de canal para escritura en 8086 y 8088.

da a la dirección de la memoria en el canal de direcciones, da salida a los datos que se van a escribir en la memoria en el canal de datos, y las señales escribir ($\overline{WR}$) a la memoria y una IO/$\overline{M}$ = 0 en el 8088 y M/$\overline{IO}$ = 1 para el 8086. Si los datos se leen de la memoria (véase la temporización simplificada para lectura en la figura 7-10), el microprocesador da salida a la dirección de la memoria en el canal de direcciones, una señal de lectura ($\overline{RD}$) de la memoria y acepta los datos por el canal de datos.

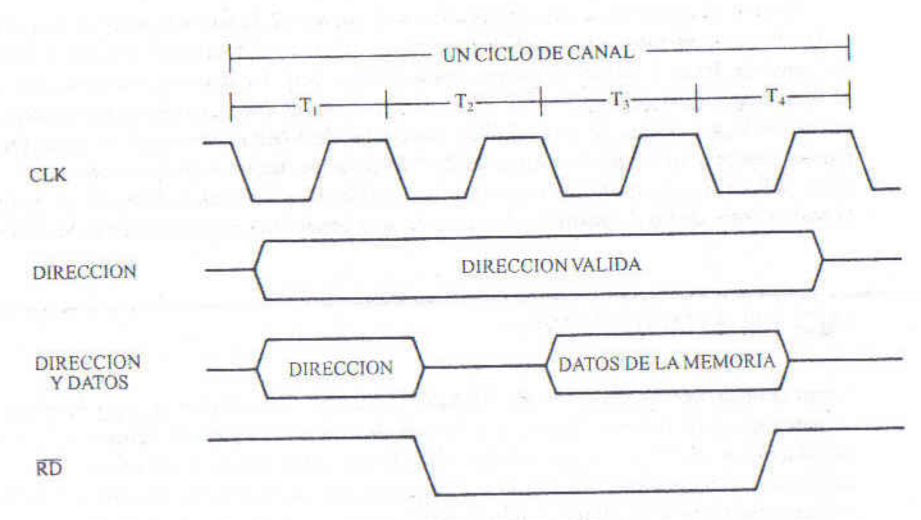


FIGURA 7–10 Ciclo simplificado de canal para lectura en 8086 y 8088.

Temporización en general

Los 8086/8088 utilizan la memoria y el espacio de E/S durante periodos de *dos ciclos del canal*. Cada ciclo es igual a cuatro periodos de reloj del sistema (estados T). Si el reloj funciona a 5 MHz (la frecuencia básica de funcionamiento de estos dos microprocesadores), entonces un ciclo de canal es de 800 ns. Esto significa que el microprocesador lee o escribe datos de la memoria o del espacio de E/S a una velocidad máxima de 1.25 millones de veces por segundo. (Debido a que poseen una cola interna, el 8086/8088 pueden ejecutar 2.5 millones de instrucciones por segundo [MIPS] en ráfagas.) Otras versiones disponibles de estos microprocesadores funcionan a velocidades de transferencia mucho más altas, debido a las frecuencias de reloj más altas.

T1. Durante el primer periodo del reloj en un ciclo de canal, que se llama T1, ocurren muchas cosas. La dirección de la memoria o del puerto de E/S se envían por el canal de direcciones y de las conexiones canal de dirección datos. (El canal de direcciones y datos están multiplexados y, durante un tiempo, contienen información de direccionamiento de la memoria y, durante otros, datos.) Durante T1 también se proporcionan las señales de control ALE, DT/R, y $IO/\overline{M}$ (8088) o $M/\overline{IO}$ (8086). La señal $\overline{IO}/M$ o $M/\overline{IO}$ indica si el canal de direcciones contiene una dirección de la memoria o el número de un puerto para un dispositivo de E/S.

T2. Durante T2, los 8086/8088 proporcionan las señales $\overline{DEN}$, $\overline{RD}$ para lectura o $\overline{WR}$ para escritura y, en el caso de escritura, los datos que se van a escribir aparecen en el canal de datos. Estos sucesos hacen que en la memoria o en E/S se inicie una lectura o una escritura. La señal $\overline{DEN}$ habilita el transceptor del canal de datos, si los hay en el sistema, a fin de que la memoria o la E/S puedan recibir los datos que se van a escribir o que el microprocesador pueda leer los datos de la memoria o de la E/S. Si ello ocurre durante un ciclo de escritura, entonces los datos se envían a la memoria o a la E/S por medio del canal de datos.

La terminal READY se muestrea al final de T2, como se ilustra en la figura 7-11. Si READY está en un nivel bajo en este momento, T3 se convierte en un estado de espera (Tw). En la sección 7-5 aparecen más detalles.

T3. Este periodo de reloj se produce a fin de dar tiempo a la memoria para accesar los datos. Si el ciclo de canal es de lectura, entonces el canal de datos se muestrea al final de T3.

T4. En T4 se desactivan todas las señales de canal en preparación para el siguiente ciclo de canal. Este es también el momento en que los 8086/8088 muestrean las terminales del canal de datos que se leen de la memoria o de E/S. Además, en este momento, el flanco de subida de WR transfiere datos a la memoria o a E/S, los cuales se activan y escriben cuando la señal WR retorna al nivel del 1 lógico.

Temporización de la lectura

En la figura 7-11 se ilustra también la temporización de la lectura para el microprocesador 8088. En el 8086 la temporización para lectura es semejante, excepto que el 8086 tiene un canal de datos de 16 en vez de 8 bits. El examen del diagrama de temporización permitirá localizar todos los eventos principales descritos para cada estado T.

El concepto más importante incluido en el diagrama de temporización de lectura es el tiempo concedido a la memoria o a E/S para leer los datos. La memoria se selecciona por su tiempo de acceso, que es el tiempo fijo que le permite el microprocesador para accesar los datos para la

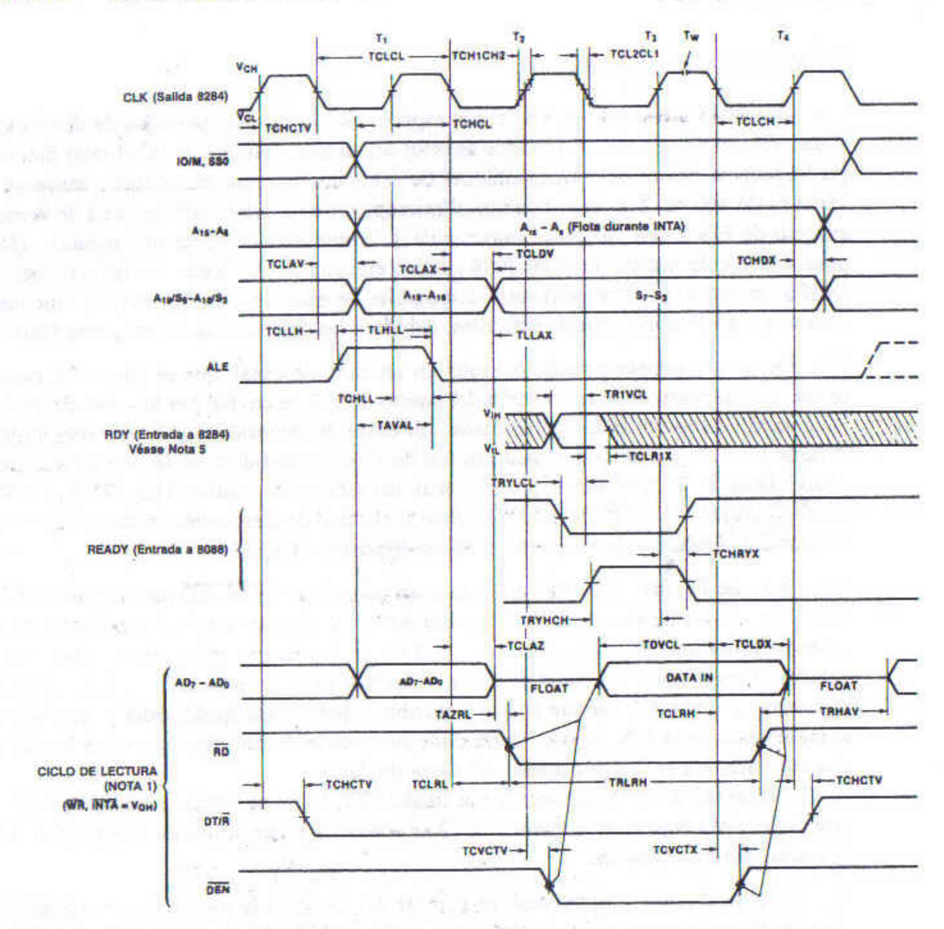


FIGURA 7-11 Sincronización de canal de 8088 en modo mínimo para una operación de lectura.

operación de lectura. Por ello, es de suma importancia que la memoria que seleccione, cumpla con las limitaciones del sistema.

Pero, el diagrama de temporización del microprocesador no está en forma directa en el tiempo de acceso a la memoria. En lugar de ello, es necesario combinar cierto número de tiempos para llegar al tiempo para acceso. Para encontrar el tiempo de acceso a la memoria en este diagrama, primero hay que localizar el punto en T3 en el cual se muestren los datos. Si se examina con detenimiento el diagrama de temporización, se notará que hay una línea que se extiende hacia abajo desde el final de T3 hasta el canal de datos. Al final de T3 es en donde el microprocesador muestrea el canal de datos.

El tiempo de acceso a la memoria se inicia cuando aparece la dirección en el canal de dirección de la memoria y continúa hasta que el microprocesador haya muestreado los datos de la memoria en T3. Transcurren tres estados de T entre estos tiempos, pero no con exactitud. (Con-

sulte la figura 7-12 para localizar los siguientes tiempos.) La dirección no aparece hasta el tiempo T_{CLAV} (110 ns si el reloj es de 5 MHz). Después del inicio de T1. Esto significa que el tiempo T_{CLAV} se debe restar de los tres estados de reloj (600 ns) que separan la aparición de la dirección (T1) y el muestreo de los datos (T3). Hay que restar también otro tiempo: el tiempo de preparación de datos (T_{DVCL}) que ocurre antes de T3. Por ello, el tiempo de acceso de memoria son tres estados de reloj, menos la suma de T_{CLAV} y T_{DVCL}. Debido a que T_{DVCL} es de 30 ns con reloj de 5 MHz, el tiempo permitido para accesar a la memoria es de sólo 460 ns (tiempo de acceso = 600 ns − 110 ns − 30 ns).

En realidad, los dispositivos de memoria seleccionados para conectarlos con los 8086 y 8088 que funcionen a 5 MHz, deberán permitir el acceso a los datos en menos de 460 ns, debido a la demora de tiempo que introducen los decodificadores de dirección y los registros de acoplamiento del sistema. Debe haber, cuando menos, un margen de 30 o 40 ns para el funcionamiento de estos circuitos. Por tanto, la velocidad de la memoria no debe ser menor de 420 ns para funcionar correctamente con los 8086/8088.

El único otro factor de temporización que puede alterar el funcionamiento de la memoria es el ancho de la señal de habilitación $\overline{RD}$. En el diagrama de temporización se da ancho de la señal RD como T_{RLRH}. El tiempo para esta señal es de 325 ns (reloj de 5 MHz), que es suficiente para casi todos los dispositivos de memoria que se fabrican, con tiempo de acceso de 400 ns o menos.

Temporización de la escritura

En la figura 7-13 se ilustra el diagrama de temporización de escritura en el microprocesador 8088. También en este caso el 8086 es muy semejante y no se necesita presentarlo en un diagrama de temporización separado.

Las diferencias principales entre la temporización para lectura y para escritura son mínimas. La señal RD se ha sustituido por $\overline{WR}$, el canal de datos contiene información para la memoria en vez de información de la memoria y DT/$\overline{R}$ es un 1 en vez de un 0 lógico durante todo el ciclo de canal.

Cuando se conectan algunos dispositivos de memoria, la temporización puede ser muy delicada entre el momento en el cual $\overline{WR}$ se hace en 1 lógico y el momento en que los datos se retiran del canal de datos. Esto ocurre porque, como se recordará, los datos se escriben en la memoria en el flanco de subida de la señal $\overline{WR}$. Según aparece en el diagrama de temporización, el periodo crítico es T_{WHDX} o sean 88 ns si el 8088 tiene reloj de 5 MHz. El tiempo de retención a veces es mucho menor y de hecho es a menudo de 0 ns para los dispositivos de memoria. El ancho de la señal $\overline{WR}$ es T_{WLWH} o 340 ns con reloj de 5 MHz. Esta velocidad, también es compatible con la mayor parte de los dispositivos de memoria que tienen un tiempo de acceso de 400 ns o menos.

7-5 LISTA Y EL ESTADO DE ESPERA

Como se mencionó antes, la entrada READY (Lista) produce estados de espera para componentes de memoria y de E/S más lentos. Un *estado de espera* (Tw) es un periodo adicional de reloj introducido entre T2 y T3 para alargar el ciclo del canal. Si se introduce un estado de espera, entonces el tiempo normal de acceso a la memoria, que es de 460 ns con reloj de 5 MHz, se alarga por un periodo de reloj (200 ns), hasta 660 ns.

FIGURA 7–12 Características de CA del 8088.

CARACTERISTICAS de ca (8088: T_A = 0°C to 70°C, V_{CC} = 5V ±10%)* (8088-2: T_A = 0°C to 70°C, V_{CC} = 5V ±5%)

REQUISITOS DE SINCRONIZACION DE UN SISTEMA DE MINIMA COMPLEJIDAD

		8088		8088-2			
Símbolo	**Parámetro**	**Min.**	**Max.**	**Min.**	**Max.**	**Unidades**	**Condiciones de la prueba**
TCLCL	Periodo de ciclo de CLK	200	500	125	500	ns	
TCLCH	Tiempo bajo de CLK	118		68		ns	
TCHCL	Tiempo alto de CLK	69		44		ns	
TCH1CH2	Tiempo de subida de CLK		10		10	ns	De 1.0 V a 3.5 V
TCL2CL1	Tiempo de caída de CLK		10		10	ns	De 3.5 V a 1.0 V
TDVCL	Datos en tiempo de ajuste	30		20		ns	
TCLDX	Datos en tiempo de retención	10		10		ns	
TR1VCL	Tiempo de ajuste de RDY en el 8284 (Ver Notas 1 y 2)	35		35		ns	
TCLR1X	Tiempo retención de RDY en el 8284 (Ver Notas 1 y 2)	0		0		ns	
TRYHCH	Tiempo de ajuste de READY en 8088	118		68		ns	
TCHRYX	Tiempo de retención de READY en 8088	30		20		ns	
TRYLCL	READY inactiva a CLK (Ver Nota 3)	−8		−8		ns	
THVCH	Tiempo de ajuste de HOLD	35		20		ns	
TINVCH	Tiempo de ajuste de INTR, NMI Y TEST Tiempo de ajuste (Véase Nota 2)	30		15		ns	
TILIH	Tiempo de subida de entrada (Excepto CLK)		20		20	ns	De 0.8 V a 2.0 V
TIHIL	Tiempo de caída de entrada ((Excepto CLK)		12		12	ns	De 2.0 V a 0.8 V

CARACTERISTICAS DE C.A. (Continúa)

RESPUESTAS DE SINCRONIZACION

		8088		8088-2			
Símbolo	**Parámetro**	**Min.**	**Max.**	**Min.**	**Max.**	**Unidades**	**Condiciones de la prueba**
TCLAV	Demora válida de dirección	10	110	10	60	ns	"C_L = 20-100 pF" para todas las salidas del 8088 además de las cargas internas
TCLAX	Tiempo retención dirección	10		10		ns	
TCLAZ	Demora en flotación de dirección	TCLAX	80	TCLAX	50	ns	
TLHLL	Ancho de ALE	TCLCH−20		TCLCH−10		ns	
TCLLH	Demora activa de ALE		80		50	ns	
TCHLL	Demora inactiva de ALE		85		55	ns	
TLLAX	Tiempo retención de dirección a ALE inactiva	TCHCL−10		TCHCL−10		ns	
TCLDV	Demora válida de datos	10	110	10	60	ns	
TCHDX	Tiempo retención de datos	10		10		ns	
TWHDX	Tiempo retención de datos después de $\overline{WR}$	TCLCH−30		TCLCH−30		ns	
TCVCTV	Demora 1 activa de control	10	110	10	70	ns	
TCHCTV	Demora 2 activa de control	10	110	10	60	ns	
TCVCTX	Demora inactiva de control	10	110	10	70	ns	
TAZRL	Flotación de dirección a READ activa	0		0		ns	
TCLRL	Demora activa de $\overline{RD}$	10	165	10	100	ns	
TCLRH	Demora inactiva de $\overline{RD}$	10	150	10	80	ns	
TRHAV	$\overline{RD}$ inactiva a siguiente dirección activa	TCLCL−45		TCLCL−40		ns	
TCLHAV	Demora válida HLDA	10	160	10	100	ns	
TRLRH	Ancho de $\overline{RD}$	2TCLCL−75		2TCLCL−50		ns	
TWLWH	Ancho de $\overline{WR}$	2TCLCL−60		2TCLCL−40		ns	
TAVAL	Dirección válida a ALE baja	TCLCH−60		TCLCH−40		ns	
TOLOH	Tiempo de subida de salida		20		20	ns	De 0.8 V a 2.0 V
TOHOL	Tiempo de caída de salida		12		12	ns	De 2.0 V a 0.8 V

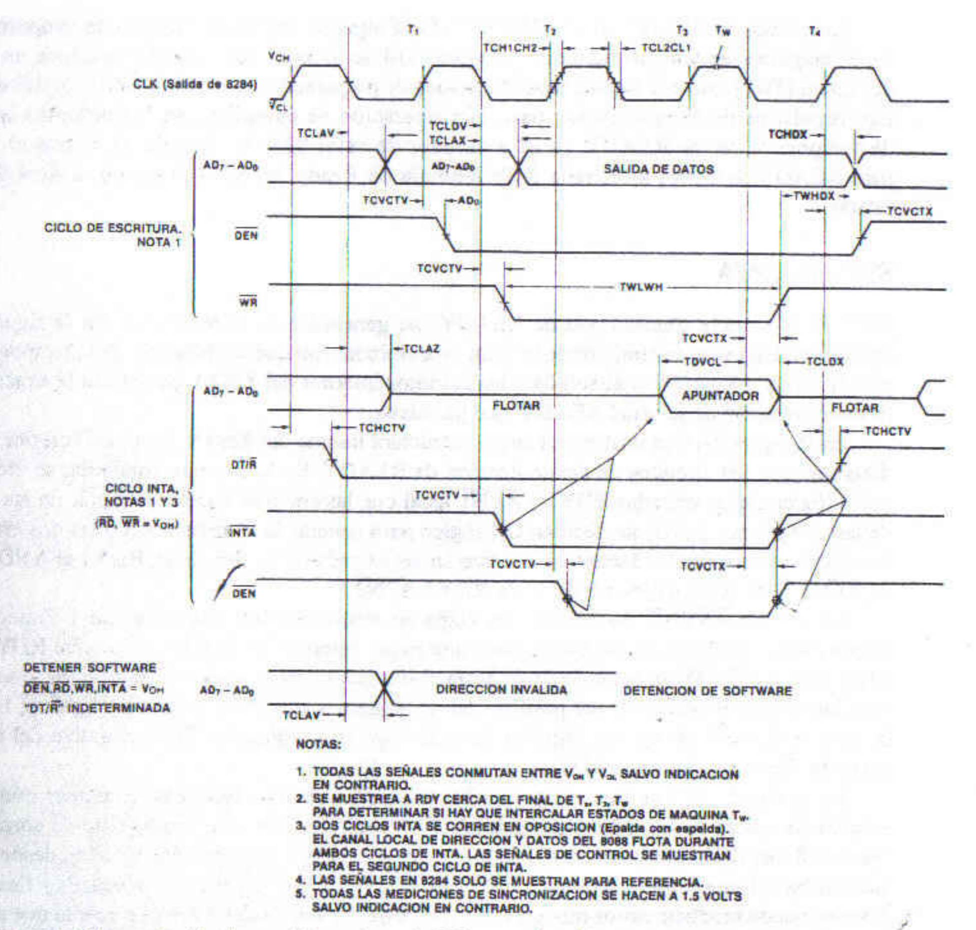


FIGURA 7–13 Sincronización de canal de escritura de 8088 en modo mínimo.

En esta sección se describen los circuitos de sincronización de la señal READY dentro del generador de reloj 8284A, se muestra la forma de introducir uno ó más estados de espera, en forma selectiva en el ciclo de canal y se examinan la entrada READY y los tiempos para sincronización que requiere.

La entrada READY

La entrada READY se muestrea al final de T2 y, de nuevo, en su caso, a la mitad de Tw. Si READY es un 0 lógico al final de T2, entonces se retrasa T3 y se inserta Tw entre T2 y T3. Después, se muestrea READY en la mitad de TW para determinar si el siguiente estado es Tw o T3. Se prueba si hay un 0 lógico en la transición de 1 a 0 del reloj al final de T2 y para ver si hay un 1 lógico en la transición de 0 a 1 del reloj en la mitad de Tw.

La entrada de READY en los 8086/8088 tiene algunos requisitos estrictos de temporización. En el diagrama de temporización de la figura 7-14 se muestra que READY ocasiona un estado de espera (Tw) junto con los tiempos requeridos de preparación y retención del reloj del sistema. Los requisitos de temporización para esta operación se cumplen con los circuitos internos de temporización de READY en el generador de reloj 8284A. Cuando se utiliza el 8284A para READY, entonces la entrada RDY (entrada de Ready del 8284A) ocurre al final de cada estado T.

RDY y el 8284A

RDY es la entrada sincronizada de READY del generador de reloj 8284A. En la figura 7-15 aparece el diagrama de temporización para esta entrada. Aunque es diferente de la temporización para la entrada READY al 8086/8088, los circuitos internos del 8284A garantizan la exactitud de la sincronización de la señal READY que producen.

En la figura 7-16 se ilustra otra vez la estructura interna del 8284A. La mitad inferior de este diagrama son los circuitos de sincronización de READY. En la extrema izquierda, se efectúa el AND lógico de las entradas RDY1 y $\overline{\text{AEN1}}$ igual que las entradas RDY2 y $\overline{\text{AEN2}}$. En las salidas de las compuertas AND, se efectúan OR lógico para generar la entrada a una o las dos etapas de sincronización. A fin de obtener un 1 lógico en las entradas a los flip-flops, RDY1 el AND lógico de $\overline{\text{AEN1}}$ debe estar activo o bien, el de RDY2 $\overline{\text{AEN2}}$.

La entrada $\overline{\text{ASYNC}}$ selecciona una etapa de sincronización cuando es un 1 lógico y dos etapas cuando es 0 lógico. Si se selecciona una etapa, entonces se impide que la señal RDY llegue a la terminal READY en los 8086/8088 hasta el siguiente flanco negativo del reloj. Si se seleccionan dos etapas, el primer flanco positivo del reloj captura a RDY en el primer flip-flop; la salida de éste se alimenta al segundo flip-flop de modo que en el siguiente flanco negativo del reloj, el segundo flip-flop, capture a RDY.

En la figura 7-17 se ilustra un circuito utilizado para introducir casi cualquier número de estados de espera en los microprocesadores 8086/8088. En este caso, un registro de corrimiento serie de 8 bits (74LS164) recorre a un 0 lógico durante uno o más periodos de reloj, desde una de sus salidas Q hasta llegar a la entrada RDY1 del 8284A. Con una conexión adecuada y flexible, el circuito puede producir varios números de estados de espera. Se debe tener en cuenta que se borra el registro de corrimiento para volver a su punto inicial. A la salida de este registro se le obliga a irse a alto cuando las terminales $\overline{\text{RD}}$, $\overline{\text{WR}}$obx e $\overline{\text{INTA}}$ son todas 1 lógico. Estas tres señales son altas hasta el estado T2, por lo cual el registro de corrimiento desplaza por primera vez cuando

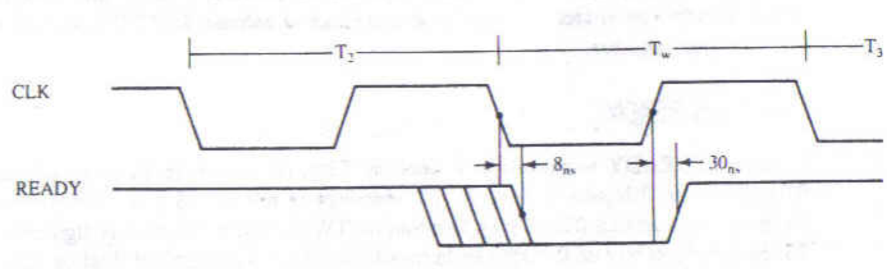


FIGURA 7–14 Sincronización de entrada READY en 8086 y 8088.

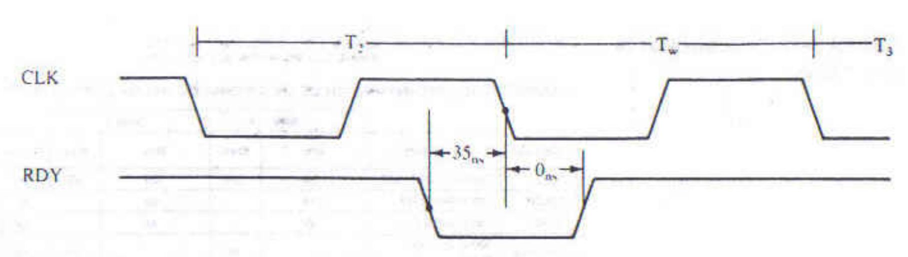


FIGURA 7-15 Sincronización de entrada RDY a 8284A.

llega el flanco positivo de T2. Si se desea una espera, entonces, la salida Q_B se conecta con la compuerta OR. Si se desean dos esperas, se conecta la salida Q_C y así sucesivamente.

Observe también en la figura 7-17 que este circuito no siempre genera estados de espera. Sólo esta capacitado por la memoria para dispositivos de memoria que requieren insertar esperas. Si la señal de selección de un dispositivo de memoria es un cero lógico, se selecciona el dispositivo; así, este circuito generará un estado de espera.

En la figura 7-18 se ilustra el diagrama de temporización del generador de estados de espera de este registro que está alambrado para insertar un estado de espera. En el diagrama de temporización también se ilustra el contenido interno de los flip-flops del registro de corrimiento, a fin de tener más detalles de su funcionamiento. En este ejemplo, se genera un estado de espera.

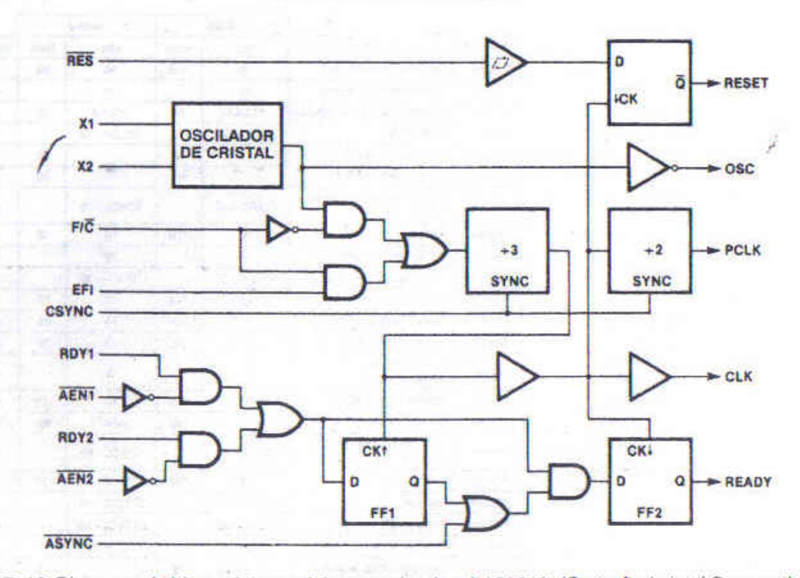


FIGURA 7-16 Diagrama de bloque interno del generador de reloj 8284A. (Cortesía de Intel Corporation.)

$\overline{CS}$ desde la memoria

RDY_1

$\overline{AEN_1}$

RDY_2

Generador de reloj 8284A

CLK

READY

0_w 1_w 2_w 3_w 4_w 5_w 6_w 7_w

Q_A Q_B Q_C Q_D Q_E Q_F Q_G Q_H

CLK 164

SI — 1

CLR

sentido del desplazamiento

READY

CLK

8088 o 8086

$\overline{RD}$

$\overline{WR}$

$\overline{INTA}$

FIGURA 7–17 Un circuito que ocasionará entre 0 y 7 estados de espera.

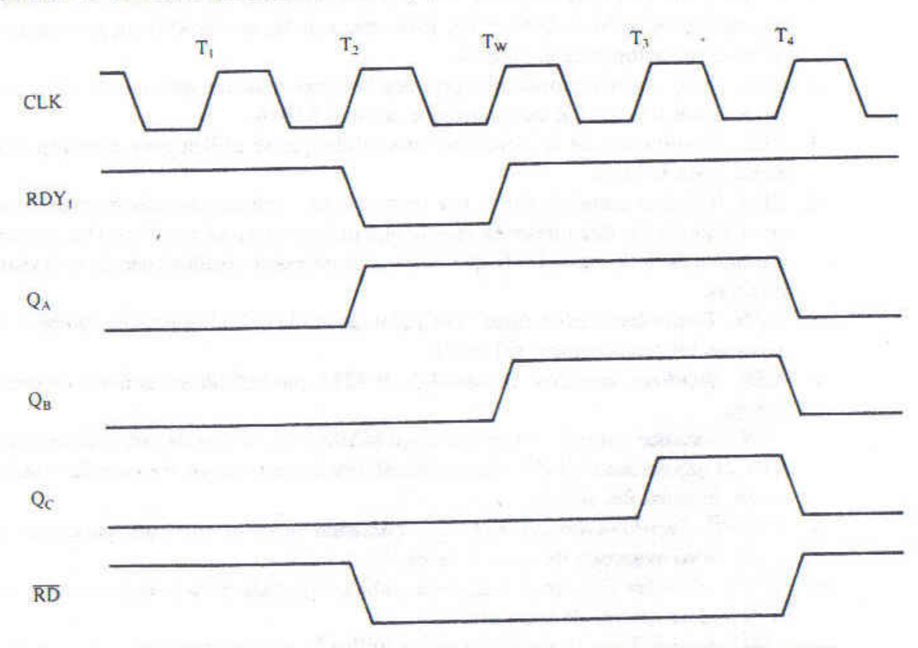


FIGURA 7–18 Sincronización de la generación de estados de espera del circuito de la figura 7-17.

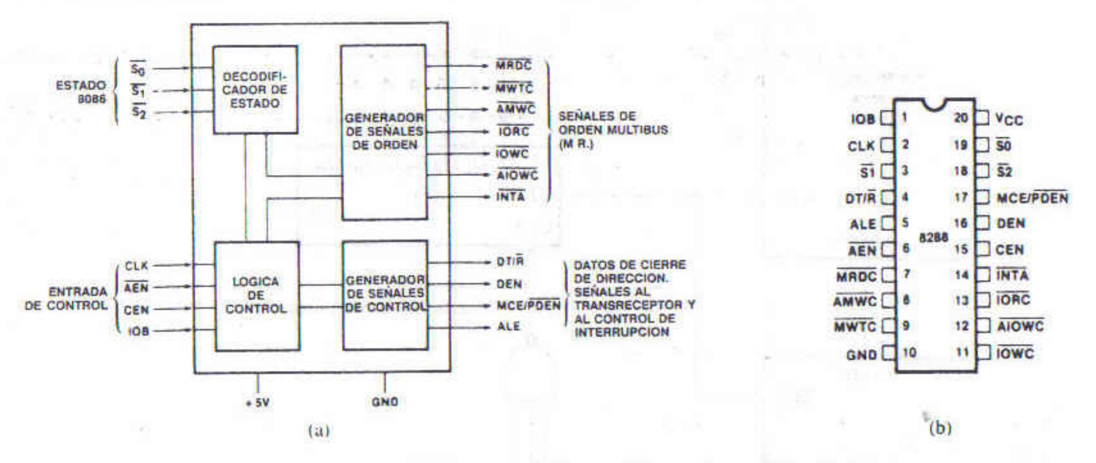


FIGURA 7–21 El controlador de canal 8288. (a) Diagrama de bloque; (b) conexiones de terminales.

Funciones de las terminales. A continuación se presenta una descripción de cada terminal del controlador de canal 8288:

1. $\overline{S2}$, $\overline{S1}$ y $\overline{S0}$. Entradas de estado: están conectadas con las terminales de salida de estado de los microprocesadores 8086/8088. Estas tres señales se decodifican para generar las señales de sincronización para el sistema.
2. CLK. *Reloj:* es una entrada que produce la temporización interna y se debe conectar con la terminal de salida CLK del generador de reloj 8284A.
3. ALE. *Habilitación de la dirección:* una salida que se utiliza para demultiplexar el canal de direcciones y datos.
4. DEN. *Habilitar canal de datos:* una terminal que controla los registros transceptores de datos en el sistema. Se debe tener en cuenta que es una terminal activa en alto que tiene polaridad opuesta a la de la señal $\overline{DEN}$ que entrega el microprocesador cuando se trabaja en el modo mínimo.
5. DT/$\overline{R}$. *Transmitir/recibir datos:* una señal de salida del 8288 para controlar el sentido de los datos en los transceptores del canal.
6. $\overline{AEN}$. *Habilitar dirección:* una entrada al 8288 que habilita las señales de control de la memoria.
7. CEN. *Habilitar control:* una entrada que habilita las señales de salida de comandos del 8288.
8. IOB. *Modo de canal de E/S:* selecciona el funcionamiento en el modo de canal de E/S o en el modo de canal del sistema.
9. $\overline{AIOWC}$. *Escribir avanzada al E/S:* un comando de salida empleado para proporcionar a la E/S una señal avanzada de control de escritura.
10. $\overline{IOWC}$. *Escribir E/S:* un comando de salida empleado para proporcionar a la E/S la señal principal de control de escritura.
11. $\overline{IORC}$. *Leer E/S:* un comando de salida utilizado para proporcionar a la E/S la señal de control de lectura.

7–6 MODO MINIMO CONTRA MODO MAXIMO

Se cuenta con dos modos de funcionamiento para los microprocesadores 8086 y 8088: el modo mínimo y el modo máximo. Para obtener el modo mínimo se conecta la terminal MN/MX a + 5.0 volts y para seleccionar el modo máximo se conecta a tierra esa terminal. Ambos modos permiten tener diferentes estructuras de control para esos microprocesadores. El modo de funcionamiento producido por el modo mínimo es similar al del 8085A, el microprocesador es de 8 bits Intel más reciente; el modo máximo es nuevo y exclusivo y se destina para utilizarlo siempre que hay un coprocesador en un sistema.

Funcionamiento en modo mínimo

El funcionamiento en modo mínimo es la forma menos costosa de hacer funcionar los microprocesadores 8086/8088. (En la figura 7-19 se ilustra un sistema de modo mínimo para el 8088.) Cuesta menos porque todas las señales para la memoria y E/S se generan dentro del microprocesador. Sus señales de control son las mismas que las del microprocesador Intel 8085A de 8 bits. Esta configuración permite utilizar periféricos del 8085A con los 8086/8088 sin ninguna dificultad especial.

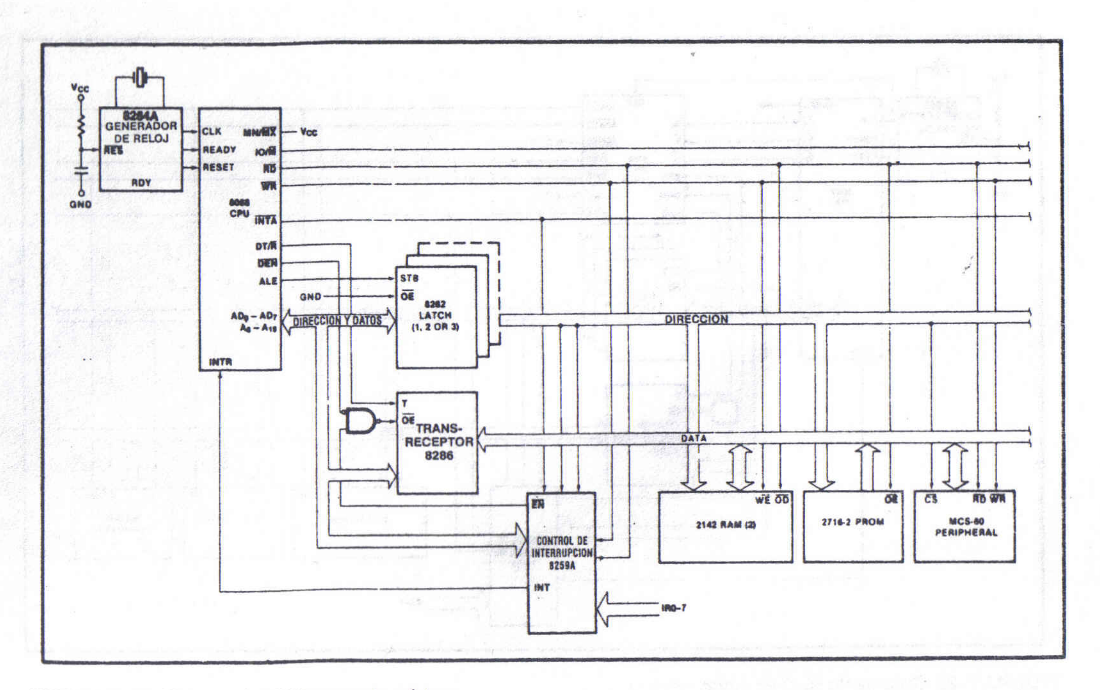


FIGURA 7–19 Sistema de 8088 en modo mínimo.

Funcionamiento en modo máximo

La diferencia entre el modo de funcionamiemto máximo y el mínimo, es que algunas de las señales de control se generarán exteriormente. Esto requiere agregar un controlador de canal externo: el 8288 (en la figura 7-20 se ilustra un sistema 8088 en modo máximo). En los 8086/8088 no hay suficientes terminales para el control del canal para el modo máximo, porque se han sustituido algunas de ellas por nuevas terminales y nuevas características. El modo máximo sólo se utiliza cuando el sistema tiene coprocesadores externos, tales como el coprocesador aritmético 8087.

Controlador de canal 8288

Un sistema para 8086/8088 que funcione en el modo máximo, debe tener un controlador de canal 8288, para producir las señales de modo mínimo que se eliminan del 8086/8088 durante el funcionamiento en modo máximo. En la figura 7-21 se ilustran el diagrama de bloque y las conexiones de las terminales del circuito del controlador de canal 8288.

Se verá que el canal de control producido por el controlador 8288 tiene señales separadas para E/S ($\overline{IORC}$ y $\overline{IOWC}$) y para memoria ($\overline{MRDC}$ y $\overline{MWTC}$). También contiene señales avanzadas de habilitación de memoria ($\overline{AMWC}$) y de E/S ($\overline{AIOWC}$) y la señal $\overline{INTA}$. Estas señales sustituyen a ALE, $\overline{WR}$, IO/$\overline{M}$, DT/$\overline{R}$, $\overline{DEN}$ e $\overline{INTA}$ que se pierden cuando se conmutan los 8086/8088 del modo mínimo al modo máximo.

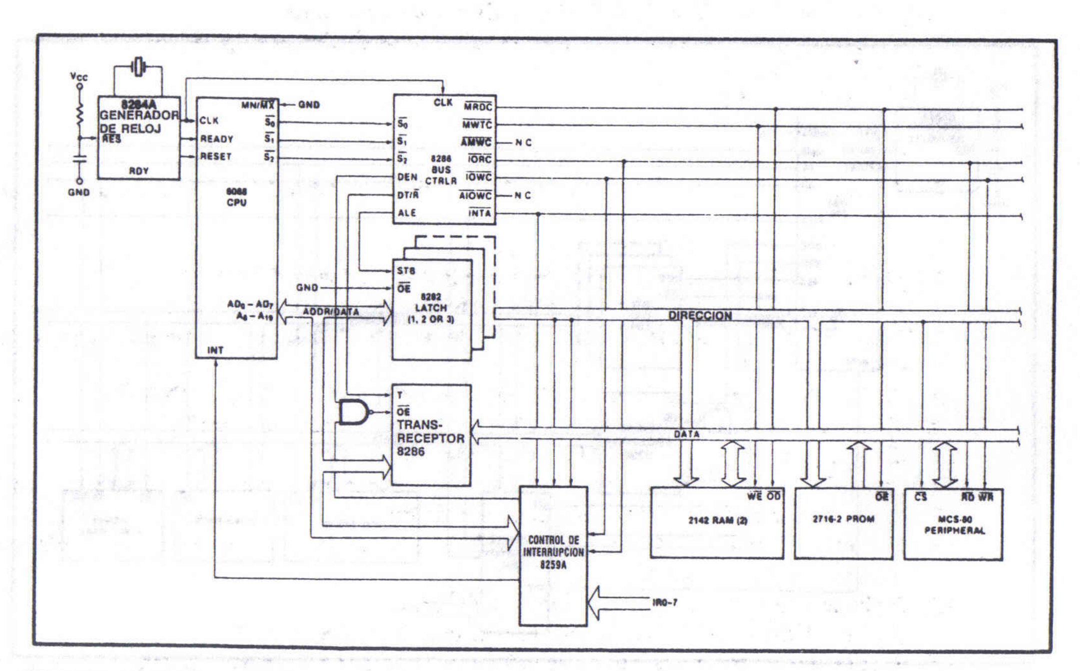


FIGURA 7–20 Sistema de 8088 en modo máximo.

12. $\overline{AMWC}$. *Señal avanzada de escritura a la memoria:* un comando utilizado para proporcionar a la memoria una señal de escritura temprana o avanzada.
13. $\overline{MWTC}$. *Escribir en la memoria:* un comando para proporcionar a la memoria una señal normal de escritura.
14. $\overline{MRDC}$. *Leer de la memoria:* un comando que proporciona a la memoria una señal de control de lectura.
15. $\overline{INTA}$. *Reconocimiento de interrupción:* una salida que reconoce una entrada de solicitud de interrupción aplicada a la terminal INTR.
16. MCE/$\overline{PDEN}$. *Cascada principal/datos periféricos:* una salida que selecciona funcionamiento en cascada para un controlador de interrupciones, si IOB está conectado a tierra y habilita a los transreceptores del canal E/S si IOB está conectado a un nivel alto.

7–7 RESUMEN

1. Las diferencias principales entre el 8086/8088 son (a) un canal de datos de 8 bits en el 8088 y de 16 bits en el 8086; (b) una terminal $\overline{SS0}$ en el 8088 en lugar de $\overline{BHE}/S_7$ en el 8086; (c) una terminal IO/$\overline{M}$ en el 8088 en lugar de una M/$\overline{IO}$ en el 8086.
2. El 8086 y el 8088 requieren un voltaje de alimentación de +5.0 volts con una tolerancia de ±10%.
3. Los 8086 y 8088 son compatibles con TTL si la cifra de inmunidad al ruido se reduce a 350 mV en vez de estar a los 400 mV usuales.
4. El 8086 y el 8088 pueden manejar: un 74XX; cinco 74LSXX; un 74SXX; diez 74ALSXX; diez 74HCXX.
5. El generador de reloj 8284A proporciona el reloj del sistema (CLK), sincronización de READY y sincronización del RESET.
6. Para obtener la frecuencia normal de funcionamiento de 5 MHz en 8086/8088, se conecta un cristal de 15 MHz en el generador de reloj 8284A. La salida PCLK contiene una señal compatible con TTL a la mitad de la frecuencia de CLK.
7. Siempre que se inicializan los 8086/8088, empiezan a ejecutar instrucciones desde la localidad de memoria FFFF0H (FFFF:0000) con la terminal de solicitud de interrupción deshabilitada.
8. Debido a que los canales para 8086/8088 son multiplexados y la mayor parte de los dispositivos de memoria y de E/S no lo son, hay que demultiplexar el canal antes de conectarlo con la memoria o con E/S. La demultiplexación se logra con un registro transparente de 8 bits, que toma su pulso de reloj de la señal ALE.
9. En un sistema grande, hay que acoplar los canales, porque los 8086/8088 sólo pueden manejar diez cargas unitarias y los sistemas grandes, a menudo, tienen más.
10. La temporización del canal es muy importante en los siguientes capítulos de este libro. Un ciclo de canal que consta de cuatro periodos de reloj, es la temporización básica del sistema. Cada ciclo de canal permite leer o escribir datos entre el microprocesador y la memoria o el sistema de E/S.
11. Un ciclo de canal se divide en cuatro estados o periodos T. El T1 lo utiliza el microprocesador para enviar la dirección a la memoria o a E/S y la señal ALE a los demultiplexores. T2 se emplea para enviar datos a la memoria para una escritura, así como para probar la terminal READY y proporcionar las señales de control $\overline{RD}$ o $\overline{WR}$. T3 proporciona tiempo a la memo-

ria para accesar a los datos y permite transferir los datos entre el microprocesador y la memoria o E/S; T4 es en donde se escriben los datos.

12. El 8086/8088 permiten a la memoria y a E/S 460 ns para accesar a los datos, cuando funcionan con un reloj de 5 MHz.
13. Los estados de espera (Tw) alargan el ciclo del canal en uno o más periodos de reloj a fin de permitir tiempo adicional para acceso a la memoria y a E/S. Para insertar los estados de espera se controla la entrada READY de los 8086/8088. La señal READY se muestrea al final de T2 y durante Tw.
14. El funcionamiento en modo mínimo es semejante al del microprocesador Intel 8085A; el funcionamiento en modo máximo es nuevo y está destinado en forma específica para el funcionamiento del coprocesador aritmético 8087.
15. El controlador de canal 8288 se debe utilizar en el modo máximo a fin de enviar las señales del canal de control a la memoria a y E/S. Esto se debe a que el funcionamiento de los 8086/8088 en el modo máximo elimina algunas de las señales de control del sistema en favor de las señales de control para los coprocesadores. El 8288 proporciona las señales de control eliminadas.

7–8 CUESTIONARIO Y PROBLEMAS

1. Enumere las diferencias entre los microprocesadores 8086 y 8088.
2. ¿Es compatible con TTL de 8086/8088? Explique su respuesta.
3. ¿Cuál es la divergencia de salida del 8086/8088 a los siguientes dispositivos:
   a. 74XXX TTL
   b. 74ALSXXX TTL
   c. 74HCXXX CMOS
   d. NMOS
4. ¿Qué información aparece en el canal de direcciones y datos del 8088 mientras ALE está activa?
5. ¿Cuál es la finalidad de los bits de estado S3 y S4?
6. ¿Qué condición indica un 0 lógico en la terminal $\overline{RD}$ del 8086/8088?
7. Describa el funcionamiento de la terminal TEST y de la instrucción WAIT.
8. Describa la señal que se aplica a la terminal de entrada CLK de los microprocesadores 8086/8088.
9. ¿Qué modo de funcionamiento se selecciona cuando se pone a tierra a MN/$\overline{MX}$?
10. ¿Qué indica la señal $\overline{WR}$ del 8086/8088 en cuanto a su funcionamiento?
11. ¿Cuándo flota ALE a su estado de alta impedancia?
12. Cuando DT/$\overline{R}$ es un 1 lógico, ¿qué condición indica en cuanto al funcionamiento del 8086/8088?
13. ¿Qué ocurre cuando la entrada HOLD a los 8086/8088 se pone a su valor de 1 lógico?
14. ¿Cuáles tres terminales de modo mínimo del 8086/8088 se decodifican para descubrir si el microprocesador está parado?
15. Explique el funcionamiento de la terminal $\overline{LOCK}$.
16. ¿Qué condiciones del 8086/8088 indican las terminales QS_1 y QS_0?
17. ¿Cuáles son las tres funciones de acondicionamiento que proporciona el generador de reloj 8284A?

18. ¿Entre cuál factor divide el generador de reloj 8284A, la frecuencia de salida del oscilador de cristal?
19. Si se conecta la terminal $F/\overline{C}$ a un valor de 1 lógico, se deshabilita el oscilador de cristal. En esta condición, ¿dónde se conecta la señal de entrada de temporización del 8284A?
20. La salida PCLK del 8284A es de ______MHz si el oscilador de cristal funciona a 14 MHz.
21. La entrada $\overline{\text{RES}}$ al 8284A se pone a un valor lógico de ______ a fin de inicializar los microprocesadores.
22. ¿En cuáles conexiones de canal se acostumbra demultiplexar en el microprocesador 8086?
23. ¿En cuáles conexiones de canal se acostumbra demultiplexar en el microprocesador 8088?
24. ¿Cuál circuito TTL integrado se utiliza a menudo para demultiplexar los canales en los 8086 y 8088?
25. ¿Cuál es la finalidad de la señal $\overline{\text{BHE}}$ demultiplexada en el microprocesador 8086?
26. ¿Por qué se requieren a menudo memorias intermedias en un sistema basado en el 8086 y 8088?
27. ¿Qué señal de 8086 y 8088 se utiliza para seleccionar el sentido de los flujos de datos en la memoria intermedia bidireccional de canal 74LS245?
28. Un ciclo de canal es igual a ______periodos de reloj.
29. Si la entrada CLK al 8086 y 8088 es de 4 MHz, ¿cuánto dura un ciclo de canal?
30. ¿Cuáles dos operaciones ocurren en 8086 y 8088 durante un ciclo de canal?
31. ¿Cuántos MIP pueden obtener 8086 u 8088 cuando funcionan con un reloj de 10 MHz?
32. Haga una breve descripción de cada uno de los estados T siguientes:
    a. T1
    b. T2
    c. T3
    d. T4
33. ¿Cuánto tiempo se permite para acceso a la memoria cuando el 8086 y 8088 funcionan con un reloj de 5 MHz?
34. ¿Cuál es el ancho de $\overline{\text{DEN}}$ si el 8088 funciona con un reloj de 5 MHz?
35. Si se pone a tierra la terminal READY, introducirá ______estados en el ciclo de canal de 8086 y 8088.
36. ¿Qué logra la entrada $\overline{\text{ASYNC}}$ al 8284A?
37. ¿Qué valores de lógica se deben aplicar a $\overline{\text{AEN1}}$ y a RDY1 para obtener un 1 lógico en la terminal READY? (Supóngase que $\overline{\text{AEN2}}$ es un 1 lógico.)
38. Mencione la diferencia entre el modo mínimo y el máximo de funcionamiento de 8086 y 8088.
39. ¿Qué función principal produce el controlador de canal 8288 cuando se utiliza con funcionamiento en modo máximo de 8086 y 8088?

CAPITULO 8

Interface con la memoria

INTRODUCCION

Cualquier sistema basado en un microprocesador, sea sencillo o complejo, tiene un sistema de memoria. La familia de microprocesadores Intel no es diferente de los otros en este aspecto.

Casi todos los sistemas tienen dos tipos principales de memoria: *memoria de sólo lectura* (ROM) y *memoria de acceso aleatorio* (RAM) o memoria de lectura y escritura. Los sistemas de memoria ROM contienen programas y datos permanentes del sistema; la memoria RAM contiene datos temporales y programas de aplicación. En este capítulo se explica cómo realizar la interface entre ambos tiempos de memoria con los microprocesadores Intel. Se demuestra la interface de memoria en un canal de datos de 8, 16 y 32 bits con el empleo de diversos tamaños de direcciones en la memoria. Esto permite que casi cualquier microprocesador pueda tener interface con cualquier sistema de memoria.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo el lector podrá:

1. Decodificar una dirección en la memoria y utilizar las salidas del decodificador para seleccionar diversos componentes de la memoria.
2. Utilizar dispositivos lógicos programables (PLD) para decodificar las direcciones en la memoria.
3. Explicar cómo se conectan la RAM y la ROM con un microprocesador.
4. Explicar cómo se pueden localizar errores en la memoria mediante la paridad.
5. Conectar la memoria con canales de datos de 8, 16 y 32 bits.
6. Explicar el funcionamiento de un controlador dinámico de RAM.
7. Conectar la RAM con el microprocesador.

8–1 DISPOSITIVOS DE MEMORIA

Antes de intentar conectar la memoria con el microprocesador, es esencial entender por completo el funcionamiento de los componentes de memoria. En esta sección, se explica la función de los tres tipos comunes de memoria: *memoria de sólo lectura* (ROM); *memoria estática de acceso aleatorio* (SRAM) y *memoria dinámica de acceso aleatorio* (DRAM).

Terminales de la memoria

Las terminales comunes para todos los dispositivos de memoria son las entradas de dirección, salidas o entradas y salidas de datos, algún tipo de entrada de selección y, cuando menos, una entrada de control utilizada para seleccionar una operación de lectura o escritura. En la figura 8-1 se presentan los dispositivos genéricos de memoria RAM y ROM.

Conexiones para la dirección. Todos los dispositivos de memoria tienen entradas de dirección que seleccionan una localidad de la memoria dentro del dispositivo. El número de terminales de dirección en un dispositivo de memoria, se determina por el número de localidades de la memoria.

En la actualidad, los dispositivos de memoria más comunes tienen entre 1K (1024) y 4M (4,194,304) localidades de memoria y en el horizonte hay dispositivos de 16M y 256M. Un dispositivo de memoria de 1K tiene 10 terminales de dirección (A_0 hasta A_9); por tanto, se requieren 10 terminales de dirección para seleccionar cualquiera de sus 1,024 localidades. Se necesita un número binario de 10 bits (1,024 combinaciones diferentes), para seleccionar una localidad cualquiera en una memoria de 1,024 localidades. Si un dispositivo de memoria tiene 11 terminales de dirección (A_0–A_{10}), tiene 2,048 (2K) localidades internas en la memoria. Por tanto, el número de localidades en la memoria se puede extrapolar con el número de terminales de dirección. Por ejemplo, un dispositivo de memoria de 4K tiene 12 conexiones para dirección; uno de 8K tiene 13 y demás. Un dispositivo que tenga 1M localidades en la memoria requiere una dirección de 20 bits (A_0–A_{19}).

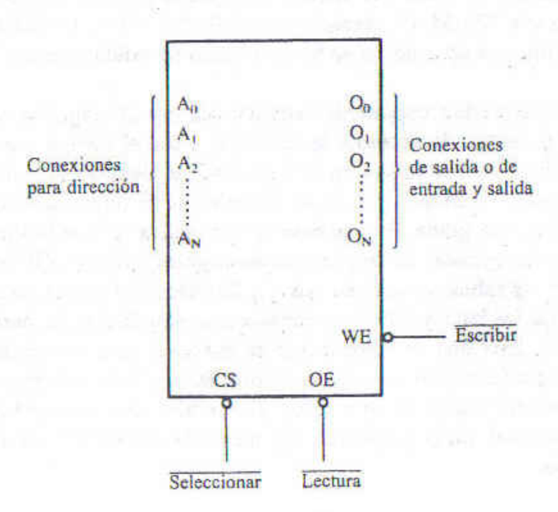


FIGURA 8–1 Un seudocomponente de memoria ilustrando las conexiones para dirección, datos y control.

Un 400H representa una sección de 1K bytes del sistema de memoria. Si se decodifica un dispositivo de memoria para empezar en una dirección 1000H y es un dispositivo de 1K, su última localidad es una dirección 103FFH, una localidad menor que 400H. Otro número hexadecimal que es importante recordar es un 10000H, porque se trata de un 4K. Un dispositivo de memoria que contiene una dirección de partida de 14000H, que es un tamaño de 4K bytes termina en la localidad 14FFFH, una localidad menos que 1000H. Un tercer número es 64K o 10000H. Una memoria que se inicia en la localidad 30000H y termina en la localidad 3FFFFFH, es de 64 bytes.

Conexiones para datos. Todos los dispositivos de memoria tienen un conjunto para salidas o entradas y salidas de datos. El dispositivo que se ilustra en la figura 8-1 tiene un conjunto común de conexiones de entrada y salida (E/S). En la actualidad, muchos dispositivos de memoria tienen terminales E/S comunes bidireccionales.

Las conexiones de datos son los puntos en los cuales se da entrada a los datos para almacenarlos o se extraen para su lectura. En este sencillo dispositivo de memoria, hay ocho conexiones de E/S, lo cual significa que el dispositivo de memoria almacena 8 bits de datos en cada localidad de la memoria. A un dispositivo de memoria de 8 bits de ancho, a veces se le llama *memoria de ancho* o *byte*. En la actualidad, aunque casi todos los dispositivos son de 8 bits de ancho no todos los dispositivos de memoria son de 8 bits de ancho. Hay algunos de 16 bits, de 4 bits o de apenas 1 bit de ancho.

En algunos catálogos, se menciona a los dispositivos como localidades de la memoria multiplicadas por los bits de cada localidad. Por ejemplo, a un dispositivo de memoria con 1K de localidades y 8 bits en cada una, el fabricante a menudo lo señala como de $1K \times 8$. Un dispositivo de memoria de $16K \times 1$ contiene 16K localidades de un bit. A las memorias, a veces, se las clasifica de acuerdo con su capacidad total en bits. Por ejemplo, a una memoria de $1K \times 8$ bits, se enumera a veces como memoria de 8K o, memoria de $64K \times 4$ se enumera como de 256K. Estas variaciones ocurren dependiendo del fabricante.

Conexiones para selección. Cada dispositivo de memoria tiene una entrada, a veces más de una, que selecciona o habilita la memoria. Esta clase de entrada se suele llamar *selección de integrado* ($\overline{CS}$) *habilitación de integrado* ($\overline{CE}$) o *selección* ($\overline{S}$). La memoria RAM por lo general, tiene una entrada $\overline{CS}$ o una ($\overline{S}$) y la ROM, cuando menos, una $\overline{CE}$. Si la entrada $\overline{CE}$, $\overline{CS}$ o $\overline{S}$) está activa (cero lógico en este caso debido a la raya en su parte superior o "testada"), el dispositivo de memoria efectúa una lectura o una escritura (si no está activa, en este caso un 1 lógico), la memoria no se puede leer ni escribir porque está apagada o deshabilitada. Si hay más de una conexión $\overline{CS}$ hay que habilitarlas todas para leer o escribir datos.

Conexiones de control. Todos los dispositivos de memoria tienen alguna forma de entrada de control. Una ROM, por lo general, tiene una sola entrada; una RAM suele tener una o dos entradas de control.

La entrada de control que se encuentra con más frecuencia en una ROM es *habilitación de salida* ($\overline{OE}$) o conexión ($\overline{G}$) de compuerta, que permiten el paso de datos de salida desde las terminales de salida de datos en la ROM. Si tanto ($\overline{OE}$) como la entrada de selección están activas, entonces se habilita la salida; si $\overline{OE}$ está inactiva, se deshabilita la salida ya que está en alta impedancia. La conexión $\overline{OE}$ habilita o deshabilita un conjunto de acopladores de tres estados, ubicados dentro de la memoria y que deben estar activos para leer los datos.

Una memoria RAM tiene una o dos entradas de control. Si hay una sola, a menudo se le llama $R/\overline{W}$. Esta terminal selecciona una operación de lectura y de escritura sólo si se selecciona la memoria con la entrada ($\overline{CS}$) de selección. Si la RAM tiene dos entradas de control, suelen estar etiquetadas $\overline{WE}$ (o $\overline{W}$) y $\overline{OE}$ (o $\overline{G}$). En este caso, $\overline{WE}$ (*habilitar escritura*) debe estar activa para efectuar una escritura en la memoria y $\overline{OE}$ debe estar activa para leer en la memoria. Cuando se cuenta con estos dos controles, no deben estar activados ambos al mismo tiempo. Si ambas entradas de control están inactivas (1 lógico), entonces no se escriben ni se leen los datos y las conexiones para datos están en su estado de alta impedancia.

Memoria ROM

En la memoria ROM (sólo lectura) se almacenan en forma permanente programas y datos que residen en el sistema y que no deben cambiar cuando se corta la corriente. La ROM tiene programación permanente, por lo cual siempre están presentes los datos, aunque se desconecte la alimentación.

La ROM está disponible en la actualidad en muchas configuraciones. El dispositivo que llamamos ROM se adquiere en grandes cantidades con los fabricantes y se programan durante su fabricación. La EPROM (*memoria de sólo lectura, programable, borrable*) que es un tipo de ROM, es de empleo más común cuando hay que cambiar la programación a menudo o cuando hay muy poca demanda de ella, para que la ROM resulte económica. Para que una ROM sea práctica, se compran, por lo general, 10,000 unidades. Una EPROM se programa en el campo en un dispositivo llamado programador de EPROM. La EPROM también se puede borrar si se la expone a luz ultravioleta de alta intensidad durante unos 20 minutos o menos, según sea el tipo de EPROM.

También hay memorias PROM disponibles, pero no se emplean mucho. La PROM (*memoria programable de sólo lectura*) también se programa en el campo, al hacer que se fundan fusibles de Nichrome o de óxido de silicio, pero una vez programada no se puede borrar.

Otro tipo, más nuevo, de memoria *mayormente de lectura* se llama memoria de "flash"[1], la cual a veces se llama también EEPROM (memoria ROM programable, eléctricamente borrable), EAROM (ROM eléctricamente alterable) o NOVRAM (ROM no volátil). Estos dispositivos de memoria se pueden borrar por acción eléctrica en el sistema, pero se requiere más tiempo para borrarlas que una RAM normal. La memoria flash se utiliza para almacenar información para ajuste de sistemas, como la tarjeta de video en la computadora. Quizá pronto sustituya a la memoria EPROM del BIOS en la computadora. Algunos sistemas incluyen una palabra de paso o contraseña en la memoria flash.

En la figura 8-2 se ilustra la memoria EPROM 2716, la cual contiene 11 entradas de dirección y 8 salidas de datos. La 2716 es una memoria de 2K × 8. En la serie 27XXX de números de pieza para las EPROM se incluyen los siguientes: 2704 (512 × 8); 2708 (1K × 8); 2716 (2K × 8); 2732 (4K × 8); 2764 (8K × 8); 27128 (16K × 8); 27256 (32K × 8); 27512 (64K × 8) y 271014 (128K × 8). Cada una de esas piezas tiene terminales para dirección, 8 conexiones para datos, una entrada $\overline{CE}$ de selección y una terminal $\overline{OE}$ de habilitación de salida.

En la figura 8-3 se muestra el diagrama de temporización de la EPROM 2716. Los datos sólo aparecen en las conexiones de salida hasta después de hacer un 0 lógico las terminales $\overline{CE}$ y $\overline{OE}$. Si ambas no son cero lógico, las conexiones de salida de datos permanecen en su estado de alta

[1] Flash memory es una marca registrada de Intel Corporation.

CONFIGURACION DE TERMINALES



SELECCION DE MODOS

MODO \ TERMINALES	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	V_{CC} (24)	SALIDAS (9-11, 13-17)
Lectura	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Deshabilitar	No importa	V_{IH}	+5	+5	High Z
No hay corriente	V_{IH}	Don't Care	+5	+5	High Z
Programa	Pulsos VIL a VIH	V_{IH}	+25	+5	D_{IN}
Verificar programa	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Inhibir programa	V_{IL}	V_{IH}	+25	+5	High Z

NOMBRES DE LAS TERMINALES

A_0–A_{10}	DIRECCIONES
PD/PGM	APAGADO Y PROGRAMA
$\overline{CS}$	SELECCIONAR INTEG.
O_0–O_7	SALIDAS

DIAGRAMA DE BLOQUE



FIGURA 8–2 Diagrama de base de la EPROM 2716 de 2K × 8. (Cortesía de Intel Corporation.)

impedancia o apagadas. Se debe tener en cuenta que la terminal Vpp debe colocarse en un 1 lógico para la EPROM lea los datos. En algunos casos la terminal Vpp está en la misma posición que la terminal $\overline{WE}$ para una SRAM. Esto puede permitir que en una sola base se conecten una EPROM o una SRAM. Un ejemplo es la EPROM 2716 y la SRAM 6116, ambas de 2K × 8, que tienen las mismas conexiones en las terminales de salida, excepto Vpp en la EPROM y $\overline{WE}$ en la SRAM.

Una información importante suministrada por el diagrama de temporización en la hoja de datos, es el tiempo de acceso a la memoria, o sea el tiempo que necesita la memoria para leer información. Como se ilustra en la figura 8-3, el tiempo (T_{acc}) de acceso a la memoria se mide desde la aparición de la dirección en las entradas de dirección, hasta la aparición de los datos en las conexiones de salida. Esto se basa en el supuesto de que la entrada $\overline{CE}$ se hace baja al mismo tiempo que las entradas de dirección están estables. Además, $\overline{OE}$ debe ser un 0 lógico para que las conexiones de salida se vuelvan activas. La velocidad básica de este EPROM es de 450 ns (recuérdese que los 8086 y 8088 que trabajan con reloj de 5 MHz, permiten hasta 460 ns para accesar a los datos). Este tipo de componente de memoria requiere estados de espera para su funcionamiento adecuado con los microprocesadores 8086 y 8088 debido a su tiempo de acceso más largo. Si no se desean estados de espera, hay disponibles versiones de EPROM con velocidad más altas a costo adicional. En la actualidad, hay memorias EPROM disponibles con tiempos de acceso de sólo 100 ns.

Características de CA

T_A = 0°C to 70°C, V_{CC}[1] = +5V ±5%, V_{PP}[2] = V_{CC} ±0.6V[3]

Símbolo	Parámetro	Límites			Unidad	Condiciones de la prueba
		Min.	Typ.[4]	Max.		
t_{ACC1}	Retardo de dirección a la salida		250	450	ns	PD/PGM = $\overline{CS}$ = V_{IL}
t_{ACC2}	PD/PGM retardo de salida		280	450	ns	$\overline{CS}$ = V_{IL}
t_{CO}	Retardo de sel. de integrado a la salida			120	ns	PD/PGM = V_{IL}
t_{PF}	PD/PGM a flotación de salida	0		100	ns	$\overline{CS}$ = V_{IL}
t_{DF}	Deshabilitar a flotación de la salida	0		100	ns	PD/PGM = V_{IL}
t_{OH}	Retención de dirección a salida	0			ns	PD/PGM = $\overline{CS}$ = V_{IL}

Capacitancia[5] T_A = 25°C, f = 1 MHz

Símbolo	Parámetro	Typ.	Max.	Unidad	Condiciones
C_{IN}	Capacitancia entrada	4	6	pF	V_{IN} = 0V
C_{OUT}	Capacitancia salida	8	12	pF	V_{OUT} = 0V

Condiciones de prueba de CA:

Carga de salida: Puerta de 1 TTL y C_L = 100 pF
Tiempos de subida y caída de entrada: ≤ 20 ns
Valor de impulsos de entrada: 0.8V a 2.2V
Valor de referencia para medir temporización:
Entradas 1V y 2V
Salidas 0.8V y 2.0 V

FORMAS DE ONDA

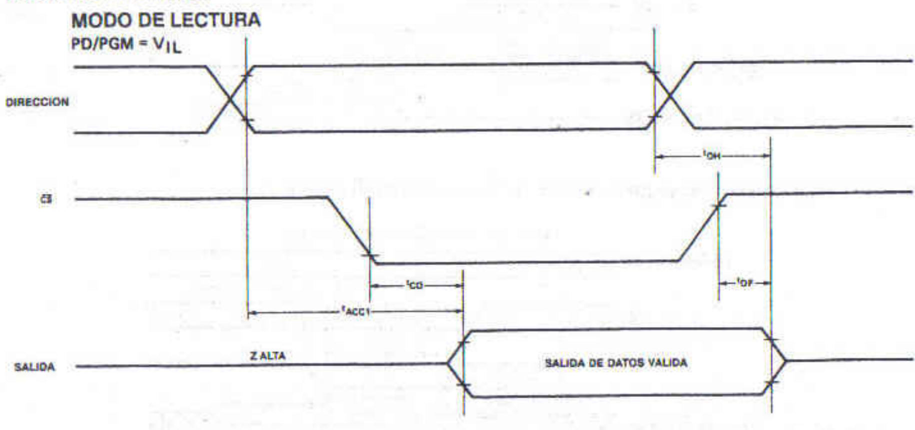


FIGURA 8–3 El diagrama de temporización y características de CA de la EPROM 2716. (Cortesía de Intel Corporation.)

Dispositivos de RAM estática (SRAM)

Las memorias RAM estáticas retienen los datos todo el tiempo que reciben alimentación (CD). Debido a que no se requiere ninguna acción especial (excepto la alimentación) para retener los datos almacenados, a este dispositivo se le llama *memoria estática*. La diferencia principal entre una ROM y una RAM es que en la RAM se escribe el funcionamiento normal mientras que la ROM se programa fuera de la computadora y, por lo general, sólo se lee. En la SRAM se almacenan datos temporales y se emplea cuando el tamaño de la memoria para lectura y escritura es pequeño. En la actualidad, una memoria pequeña es una de menos de 1M byte.

FIGURA 8-4 Diagrama de base de la RAM estática, TMS4016 (SRAM) DE 2K × 8. (Cortesía de Texas Instruments, Inc.)

TMS4016, PAQUETE NL
(vista superior)

A7	1	24	V_{CC}
A6	2	23	A8
A5	3	22	A9
A4	4	21	$\overline{W}$
A3	5	20	$\overline{G}$
A2	6	19	A10
A1	7	18	$\overline{S}$
A0	8	17	DQ8
DQ1	9	16	DQ7
DQ2	10	15	DQ6
DQ3	11	14	DQ5
V_{SS}	12	13	DQ4

NOMENCLATURA DE TERMINALES	
A0 – A10	Direcciones
DQ1 – DQ8	Entrada y salida de datos
$\overline{G}$	Habilitar salida
$\overline{S}$	Seleccionar circuito integrado
V_{CC}	Corriente de +5.0V
V_{SS}	Tierra
$\overline{W}$	Habilitar escritura

En la figura 8-4 se ilustra la SRAM 4016, la cual es una memoria de lectura y escritura de 2K × 8; tiene 11 entradas de dirección y 8 conexiones para entrada y salida de datos.

Las entradas de control en la RAM son un poco distintas a las que se describieron antes. La terminal $\overline{OE}$ es ahora $\overline{G}$, la terminal $\overline{CS}$, $\overline{S}$ y la terminal $\overline{WE}$, $\overline{W}$. A pesar de estas alteraciones en las designaciones, las terminales de control funcionan exactamente como las ya descritas. Otros fabricantes producen esta SRAM tan popular con los números de pieza 2016 y 6116.

En la figura 8-5 se presenta el diagrama de temporización de la SRAM 4016. Como lo muestra el ciclo de temporización para lectura, el tiempo de acceso es $t_a(A)$. En la más lenta de las 4016 este tiempo es de 250 ns, pero es lo bastante veloz para conectar con un 8086 o un 8088 que trabajen a 5 MHz sin estados de espera. También en este caso es importante recordar que se debe comprobar el tiempo de acceso para determinar la compatibilidad de los componentes de la memoria con el microprocesador.

En la figura 8-6 se ilustran las conexiones de terminales de la RAM estática 62256, de 32K × 8. Esta memoria consta de un circuito integrado con 28 terminales y está disponible con tiempos de acceso de 120 ns o de 150 ns. Otras memorias SRAM en uso son en tamaños de 8K × 8 y de 128K × 8.

Memoria RAM dinámica (DRAM)

La RAM estática más grande que hay en la actualidad es de 128K × 8. Por otra parte, las RAM dinámicas están disponibles en tamaños mucho más grandes, hasta de 16M × 1. En todos los demás aspectos, la DRAM, en esencia, es lo mismo que la SRAM excepto que retiene datos sólo durante 2 o 4 ms en un capacitor integrado. Después de 2 o de 4 ms, hay que "*refrescar*" por

características eléctricas en el intervalo recomendada de temperatura de funcionamiento al aire libre (salvo indicación en contrario)

PARAMETRO		CONDICIONES DE LA PRUEBA		MIN	TIPICA†	MAX	UNIDAD
V_{OH}	Alto voltaje	$I_{OH} = -1$ mA,	$V_{CC} = 4.5$ V	2.4			V
V_{OL}	Voltaje de línea	$I_{OL} = 2.1$ mA,	$V_{CC} = 4.5$ V			0.4	V
I_I	Corriente de entrada	$V_I = 0$ V to 5.5 V				10	µA
I_{OZ}	Corriente de salida en estado inactivo	$\overline{S}$ or $\overline{G}$ at 2 V or $\overline{W}$ at 0.8 V, $V_O = 0$ V to 5.5 V				10	µA
I_{CC}	Suministro de corriente de V_{CC}	$I_O = 0$ mA, $T_A = 0°C$ (peor caso)	$V_{CC} = 5.5$ V,		40	70	mA
C_i	Capacitancia de entrada	$V_I = 0$ V,	f = 1 MHz			8	pF
C_O	Capacitancia de salida	$V_O = 0$ V,	f = 1 MHz			12	pF

†A todos los valores típicos son con Vcc= 5V, TA =25°C.

requisitos de temporización en la gama recomendada de voltaje de entrada y la gama de temperatura de funcionamiento al aire libre

PARAMETRO		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIDAD
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{c(rd)}$	Tiempo de ciclo de lectura	120		150		200		250		ns
$t_{c(wr)}$	Tiempo de ciclo de escritura	120		150		200		250		ns
$t_{w(W)}$	Ancho de impulso de lectura	60		80		100		120		ns
$t_{su(A)}$	Tiempo de ajuste de dirección	20		20		20		20		ns
$t_{su(S)}$	Tiempo de ajuste de selección de integrado	60		80		100		120		ns
$t_{su(D)}$	Tiempo de ajuste de datos	50		60		80		100		ns
$t_{h(A)}$	Tiempo de retención de dirección	0		0		0		0		ns
$t_{h(D)}$	Tiempo de retención de datos	5		10		10		10		ns

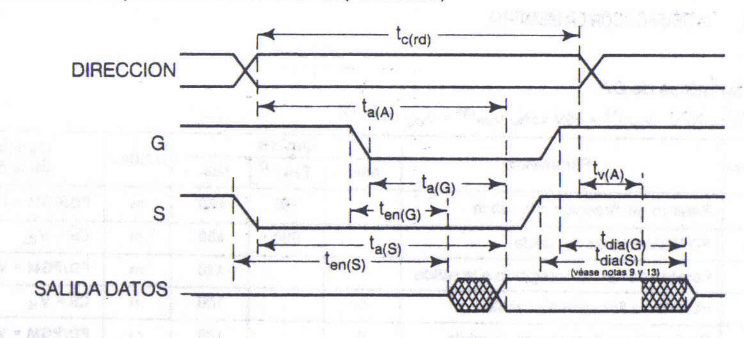
características de conmutación en la gama recomendada de voltaje $T_A = 0°$ a $70°C$

PARAMETRO		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIDAD
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{a(A)}$	Tiempo de acceso desde la dirección		120		150		200		250	ns
$t_{a(S)}$	Tiempo de acceso desde selección de integrado		60		75		100		120	ns
$t_{a(G)}$	Tiempo de acceso desde habilitar salida baja		50		60		80		100	ns
$t_{v(A)}$	Tiempo de salida válido después del cambio de dirección	10		15		15		15		ns
$t_{dis(S)}$	Tiempo de deshabilitación salida después de selección de integrado alta		40		50		60		80	ns
$t_{dis(G)}$	Tiempo de deshabilitación de la salida después de habilitar salida alta		40		50		60		80	ns
$t_{dis(W)}$	Tiempo de deshabilitación de la salida después de habilitar escritura baja		50		60		60		80	ns
$t_{en(S)}$	Tiempo de habilitación de la salida después de selecc. integrado baja	5		5		10		10		ns
$t_{en(G)}$	Tiempo de habilitación salida después de habilitación salida baja	5		5		10		10		ns
$t_{en(W)}$	Tiempo de habilitación salida después habilitación escritura alta	5		5		10		10		ns

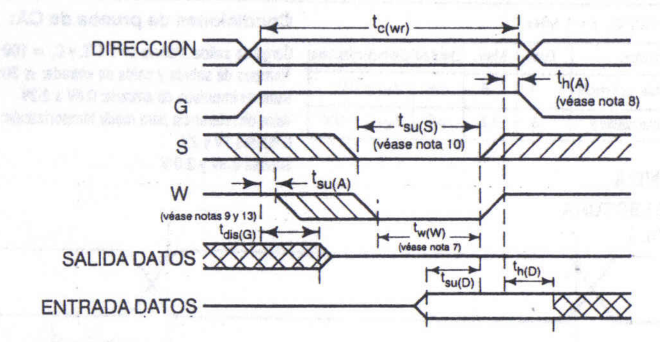
NOTAS: 3. $C_L = 100\ t_{dis}/t_{en}$ pF para todas las mediciones, excepto t_{disw} y $t_{en(W)}$.
$C_L = 5$ pF para $t_{dis(W)}$ y $t_{en(W)}$.
4. Los parámetros t_{dis} y t_{en} se muestrean y no se prueban 100%.

FIGURA 8–5 (a) Las características de CA de la SRAM TMS4016. (b) Diagrama de temporización de la SRAM TMS4016. (Cortesía de Texas Instruments, Inc.)

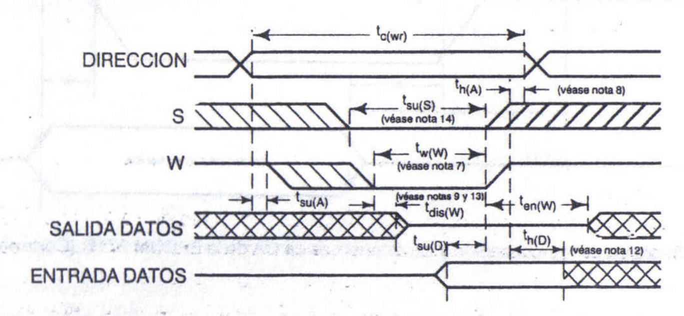
forma de onda de temporización del ciclo de lectura (véase nota 5)



forma de onda de temporización del ciclo No. 1 de escritura (véase nota 6)



forma de onda de temporización del ciclo No. 2 de escritura (véase nota 6 y 11)



NOTAS:

5. $\overline{W}$ es ciclo de lectura alto.
6. $\overline{W}$ debe estar alta durante todas las transiciones de dirección.
7. Ocurre una escritura durante $\overline{S}$ baja y $\overline{W}$ baja simultáneas.
8. $t_{h(A)}$ se mide desde lo que ocurra primero de que $\overline{S}$ o $\overline{W}$ vaya a altos hasta el final del ciclo de escritura.
9. Durante este periodo las terminales E/S están en estado de salida, por lo cual no se deben aplicar las señales de entrada de la fase opuesta a las salidas.
10. Si la transición lenta ocurre al mismo tiempo que las transiciones bajas de $\overline{W}$, o después de la transición a $\overline{W}$ la salida permanece en estado de alta impedancia.
11. G está baja en forma continua ($G = V_{IL}$).
12. Si $\overline{S}$ está baja durante este periodo, las terminales E/S están en estado de salida, por lo cual no se deben aplicar las señales de entrada de la fase opuesta a las salidas.
13. La transición se mide a ±200 mV del voltaje en estado estable.
14. Si la transición de $\overline{S}$ a baja ocurre antes de la transición de W a baja, no se deben aplicar las señales de entrada de datos de la fase opuesta a las salidas mientras dure $t_{dis(W)}$ después de la transición de $\overline{W}$ a baja.

FIGURA 8–5 *(continuación)*

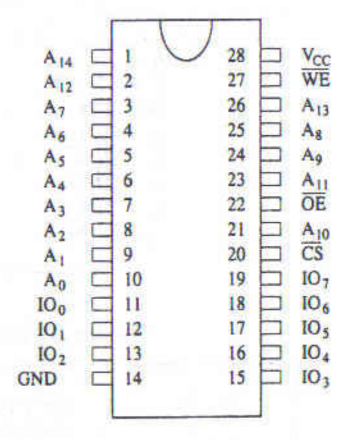


FIGURA 8–6 Diagrama de base de la RAM estática 62256 de 32K × 8.

FUNCION DE LAS TERMINALES

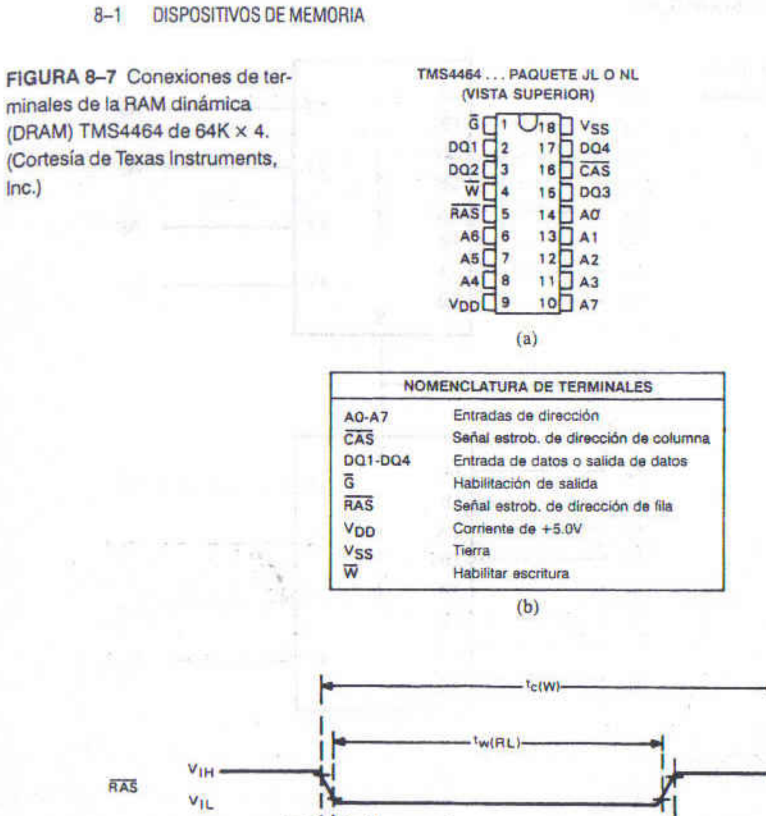
A_0 - A_{14}	Direcciones
IO_0 - IO_7	Terminales de datos
$\overline{CS}$	Selección de integrado
$\overline{OE}$	Habilitar salida
$\overline{WE}$	Habilitar escritura
V_{CC}	Corriente de +5.0 V
GND	Tierra

completo el contenido de la memoria, porque los capacitores, que almacenan un 1 lógico o un 0 lógico, pierden sus cargas.

En lugar de necesitar la tarea casi imposible de leer el contenido de cada localidad de memoria con un programa y, luego, volver a escribirlas, el fabricante ha hecho que la construcción interna de la DRAM, en su versión de 64K × 1, permita refrescarla con 256 lecturas en un intervalo de 4 ms. Esto también ocurre durante un ciclo de escritura, lectura o uno especial. En la sección 8-6 aparece mucha más información para refrescar las DRAM.

Otra desventaja de la memoria DRAM es que requiere tal cantidad de terminales para dirección, que el fabricante ha multiplexado las entradas de dirección. En la figura 8-7 se ilustra una DRAM, No. TMS4464, de 64K × 4. Se verá que sólo tiene 8 entradas de dirección, cuando debería tener 16, o sea el número requerido para direccionar 64K localidades en la memoria. La única forma en que los 16 bits de dirección se puedan agrupar en 8 terminales de dirección, es en dos etapas de 8 bits. Esta operación requiere dos terminales especiales llamadas *señal estroboscópica de dirección de columna* ($\overline{CAS}$) y *señal estroboscópica de dirección de renglón* ($\overline{RAS}$). Primero, se presentan A_0–A_7 en las terminales de dirección y se habilitan con la señal estroboscópica $\overline{RAS}$ en un registro interno como la dirección de renglón. Luego, se presentan los bits A_8–A_{15} en las mismas ocho entradas de dirección y con la señal estroboscópica $\overline{CAS}$ habilitan un registro

FIGURA 8-7 Conexiones de terminales de la RAM dinámica (DRAM) TMS4464 de 64K × 4. (Cortesía de Texas Instruments, Inc.)



(a)

NOMENCLATURA DE TERMINALES	
A0-A7	Entradas de dirección
$\overline{CAS}$	Señal estrob. de dirección de columna
DQ1-DQ4	Entrada de datos o salida de datos
$\overline{G}$	Habilitación de salida
$\overline{RAS}$	Señal estrob. de dirección de fila
V_{DD}	Corriente de +5.0V
V_{SS}	Tierra
$\overline{W}$	Habilitar escritura

(b)

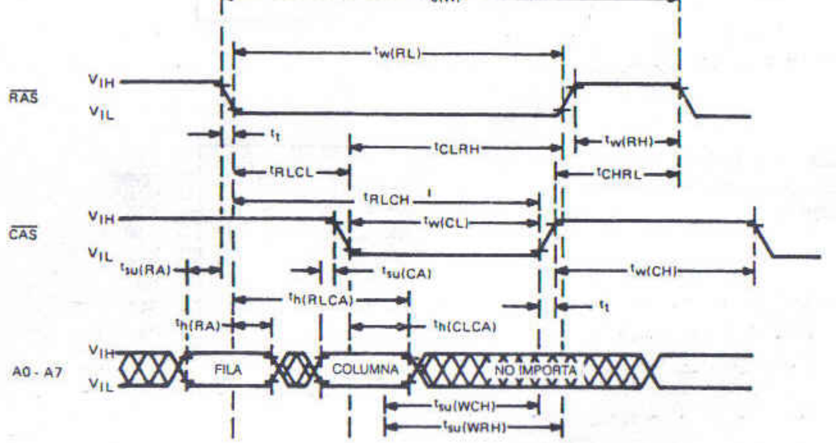


FIGURA 8-8 Temporización de $\overline{RAS}$, $\overline{CAS}$ y entrada de dirección para la DRAM TMS4464. (Cortesía de Texas Instruments, Incorporated.)

interno como la dirección de columna (en la figura 8-8 aparece esta temporización). La dirección de 16 bits de los registros internos direcciona el contenido de una de las localidades de 4 bits en la memoria.

En la figura 8-9 se ilustra un conjunto de multiplexores utilizados para la selección estroboscópica de las direcciones de columna y de renglón, en las ocho entradas de dirección de un par de

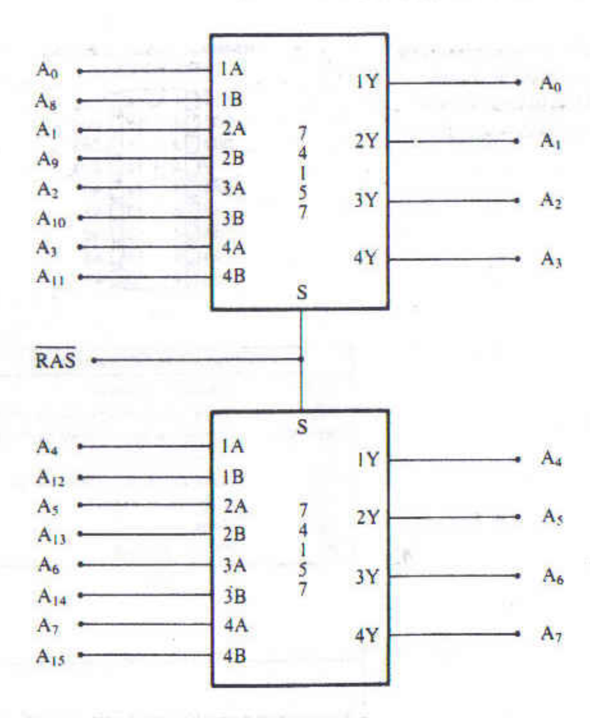


FIGURA 8–9 Multiplexor de dirección para la DRAM TMS4464.

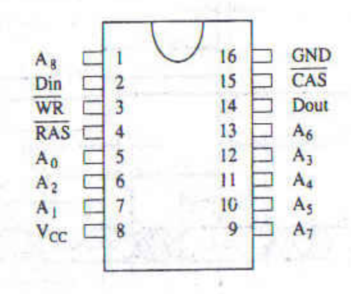
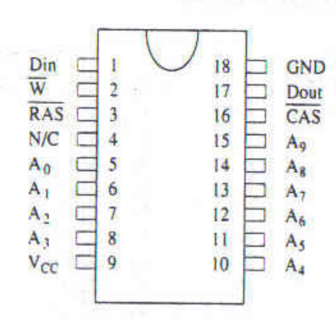


FIGURA 8–10 La RAM dinámica 41256 organizada como memoria de 256K × 1.

FUNCION DE LAS TERMINALES

A_0 - A_8	Direcciones
Din	Entrada de datos
Dout	Salida de datos
$\overline{CAS}$	Señal estrob. de dirección de columna
$\overline{RAS}$	Señal estrob. de dirección de fila
$\overline{WR}$	Habilitar escritura
V_{CC}	Corriente de +5.0V
GND	Tierra

FIGURA 8-11 La DRAM de 1M × 1.



FUNCION DE LAS TERMINALES

A_0 - A_9	Direcciones
Din	Entrada de datos
Dout	Salida de datos
$\overline{CAS}$	Señal estrob. de dirección de columna
$\overline{RAS}$	Señal estrob. de dirección de fila
$\overline{W}$	Habilitar escritura
V_{CC}	Corriente de +5.0V
GND	Tierra

DRAM TMS4464. En este caso, la $\overline{RAS}$ no sólo da la selección estroboscópica de la nueva dirección en la DRAM, sino que también cambia la dirección aplicada a las entradas de dirección. Esto es posible, debido al largo tiempo de propagación de los multiplexores. Cuando $\overline{RAS}$ es un 1 lógico, las entradas B se conectan con las salidas Y de los multiplexores y cuando $\overline{RAS}$ es un 0 lógico, las entradas A se conectan con las salidas Y. Debido a que el registro interno de dirección de renglón se dispara por flanco, captura la dirección del renglón antes que la dirección en las entradas cambie a la dirección de la columna. En la sección 8-6 se dan más detalles de la interface de DRAM de la DRAM.

Al igual que con la SRAM, la terminal $\overline{W}$ escribe datos en la DRAM y la terminal $\overline{G}$ habilita las conexiones de salida para una operación de lectura. ($\overline{W}$ sustituye a $\overline{WE}$ y $\overline{G}$ sustituye a $\overline{OE}$). En la figura 8-10 se ilustran las conexiones en las terminales de la RAM dinámica 41256. Esta memoria está organizada como de 256K × 1 y requiere apenas 70 ns para accesar a los datos.

En fechas más recientes ya está disponible una DRAM más grande organizada como memoria de 1M × 1. En el futuro habrá las memorias de 16M × 1 y 256M × 1 que se encuentran en las etapas de planeación. En la figura 8-11 se ilustra el diagrama de base de la memoria 1M × 1, la TMX4C1024 de Texas Instruments, la cual, a veces, lleva el número 511000P.

8–2 DECODIFICACION DE LA DIRECCION

A fin de poder conectar un dispositivo de memoria con el microprocesador, es necesario decodificar la dirección en él para que sea una sección o división exclusiva del mapa de memoria. Al no

tener un decodificador de dirección, sólo se puede conectar un dispositivo de memoria en el microprocesador, lo cual casi lo inutilizaría. En esta sección se describen algunas de las técnicas más comunes para decodificar direcciones así como los decodificadores que se encuentran en muchos sistemas.

¿Por qué decodificar la memoria?

Cuando el microprocesador 8088 se compara con la EPROM 2716 se ve con claridad la gran diferencia en el número de conexiones para direcciones: la EPROM tiene 11 conexiones y el microprocesador tiene 20. Esto significa que el microprocesador da salida a una dirección de memoria de 20 bits cada vez que lee o escribe datos. Dado que la EPROM tiene sólo 11 entradas para dirección, hay una discordancia que se debe corregir en alguna forma. Si sólo 11 de las terminales de dirección del 8088 están conectadas con la memoria, entonces éste sólo verá 2K bytes de memoria en lugar de los 1M bytes que "espera" que contenga la memoria. El decodificador se utiliza para hacer acoplar el microprocesador con el componente de memoria.

Decodificador sencillo por puerta NAND

Las conexiones de direcciones A_{10}–A_0 del 8088 están conectadas con las entradas de dirección (A_{10} hasta A_0) de la EPROM y las nueve terminales restantes de dirección (A_{19} hasta A_{11}) están conectadas con las entradas de un decodificador de compuerta NAND (figura 8-12). El decodificador selecciona la EPROM entre una de las muchas secciones de 2K bytes de todo el intervalo de 1M byte de direcciones del microprocesador 8088.

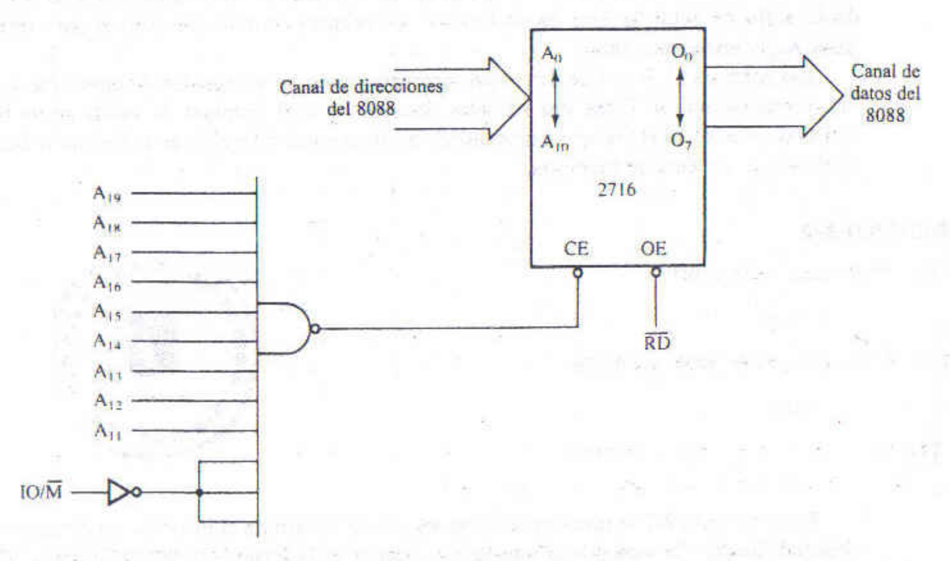


FIGURA 8–12 Un decodificador sencillo con compuerta NAND utilizado para seleccionar un componente de memoria EPROM 2716 para las direcciones FF800H hasta FFFFFH en la memoria.

En este circuito, una sola compuerta NAND decodifica la dirección en la memoria. La salida de esta compuerta NAND es un 0 lógico siempre que las terminales de dirección del 8088 conectadas con sus entradas A_{19}–A_{11} sean todas 1 lógico. La salida baja activa, de 0 lógico del decodificador citado está conectada con la entrada $\overline{CE}$ la cual habilita la EPROM. Recuérdese que siempre que $\overline{CE}$ es un 0 lógico. los datos se leerán en la EPROM sólo si $\overline{OE}$ es también un 0 lógico. La terminal $\overline{OE}$ se activa mediante la señal $\overline{RD}$ del 8088 o por la señal $\overline{MRDC}$ (control de lectura de la memoria) de otros componentes de la "familia".

Si la dirección binaria de 20 bits decodificada por el decodificador NAND, se escribe de modo que los 9 bits que están más a la izquierda sean 1 lógico y los 11 bits de la derecha son "no importa"(X), se puede determinar el intervalo real de dirección en EPROM. (Un no importa es un 1 lógico o un 0 lógico, lo que corresponda.)

EJEMPLO 8–1

```
1111 1111 1XXX XXXX XXXX

        o bien

1111 1111 1000 0000 0000 = FF800H

        hasta

1111 1111 1111 1111 1111 = FFFFFH
```

En el ejemplo 8-1 se ilustra la forma en que el intervalo de dirección para esta EPROM se determina al escribir los bits de dirección decodificados en el exterior (A_{19} hasta A_{11}) y a los bits de dirección de la EPROM (A_{10} hasta A_0) se los califica de no importa. Como se muestra en el ejemplo, los no importa se escriben primero como ceros, para localizar la dirección más baja y, luego, como 1 para encontrar la dirección más alta. En el ejemplo también se muestran estos límites como direcciones hexadecimales. En este caso, la EPROM de 2K se decodifica en las localidades FF800H hasta FFFFFH en la memoria. Se verá que es una sección de 2K bytes de la memoria y que aquí se encuentra la localidad de inicialización del 8086 y 8088, que es el sitio más adecuado para una EPROM.

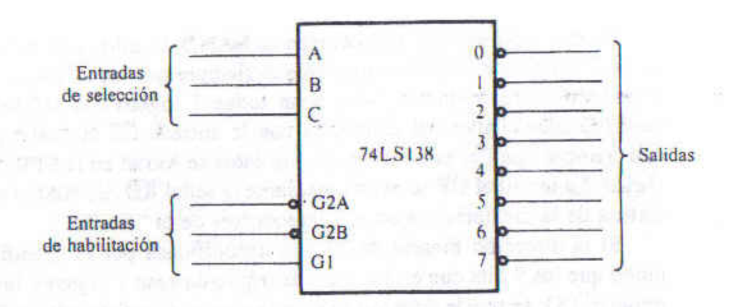
Aunque este ejemplo sirve para describir la decodificación, las compuertas NAND rara vez se utilizan para decodificar memorias, porque cada dispositivo de memoria requiere su propio decodificador por compuerta NAND. Debido al excesivo costo de este decodificador, esta opción requiere encontrar un componente alterno.

Decodificador de 3 a 8 líneas (74LS138)

Uno de los decodificadores integrados más comunes, aunque no el único, que se encuentra en muchos sistemas basados en un microprocesador, es el decodificador de 3 a 8 líneas 74LS138. En la figura 8-13 se muestran este decodificador y su tabla de la verdad.

La tabla de la verdad muestra que sólo una de las ocho salidas baja (cero lógico) en cualquier momento. Para que cualquiera de las salidas del decodificador sea baja, deben estar activas las tres entradas de habilitación; $\overline{G2A}$, $\overline{G2B}$ y $\overline{G1}$; para ello, las entradas $\overline{G2A}$ y $\overline{G2B}$ deben ser bajas (0 lógico) y G1 debe ser alta (1 lógico).

FIGURA 8–13 El decodificador 74LS138 de 3 a 8 líneas y su tabla de función.



Entradas						Salidas							
Habilitar			Seleccionar										
$\overline{G2A}$	$\overline{G2B}$	G1	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

Una vez que se habilita el 74LS138, las entradas (C, B y A) de dirección seleccionan cuál terminal de salida baja. Imagínense ocho entradas $\overline{CE}$ a la EPROM conectadas con las ocho salidas del decodificador. Se trata de un dispositivo muy potente porque selecciona hasta ocho diferentes dispositivos de memoria.

Muestra de un circuito decodificador. Se verá que las salidas del decodificador de la figura 8-14 están conectadas con ocho diferentes dispositivos de memoria EPROM 2764. El decodificador selecciona ocho bloques de memoria de 8K bytes, para un total de 64K bytes de memoria. También se ilustra el intervalo de direcciones de cada dispositivo y las conexiones comunes en éste. Se verá que todas las conexiones de direcciones del 8088 están conectadas a este circuito. También se apreciará que las salidas del decodificador están conectadas con las entradas $\overline{CE}$ de las EPROM y la señal $\overline{RD}$ del 8088 está conectada con las entradas $\overline{OE}$ de las EPROM. Esto permite habilitar sólo a la EPROM seleccionada y enviar los datos al microprocesador por el canal de datos, siempre que $\overline{RD}$ se convierte en un 0 lógico.

En este circuito, hay una compuerta NAND de 3 entradas conectada con los bits de dirección A_{19} hasta A_{17}. Cuando las tres entradas de dirección están altas, la salida de compuerta NAND

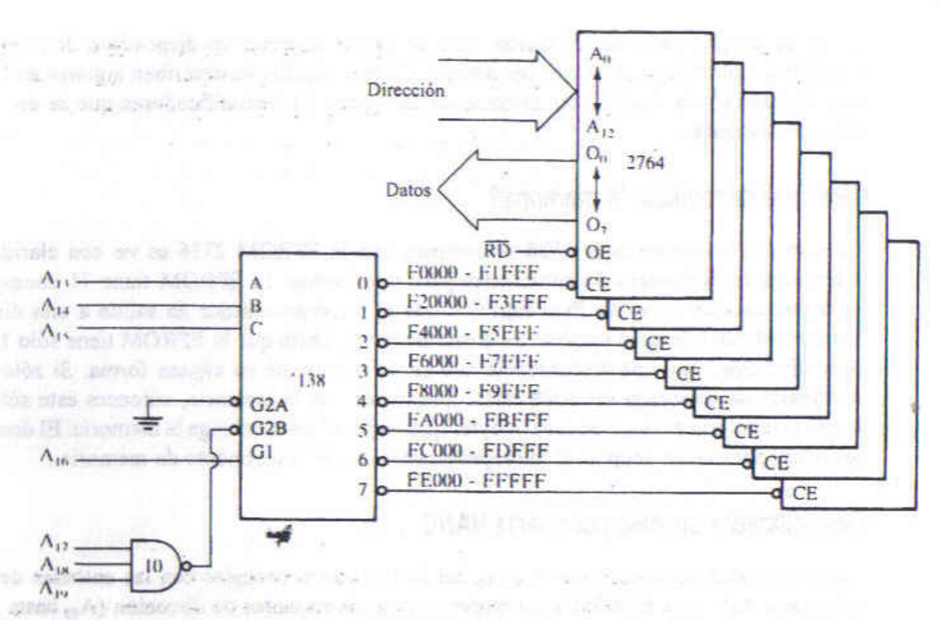


FIGURA 8–14 Un circuito en que se emplean ocho EPROM 2764 para una sección de memoria de 64K × 8 en un sistema basado en microprocesador. Las direcciones seleccionadas en este circuito son F0000H - FFFFFH.

baja y habilita la entrada $\overline{G2B}$ del 74LS138. La entrada G1 tiene conexión directa con A_{16}. Es decir, a fin de habilitar este decodificador, las primeras cuatro conexiones para dirección A_{19} hasta A_{16}) deben estar altas.

Las entradas C, B y A de dirección se conectan con las terminales de dirección A_{15} hasta A_{16} del microprocesador. Estas tres entradas determinan cuál terminal de salida se va baja y cual EPROM se selecciona siempre que el 8088 da salida a una dirección en la memoria dentro de este intervalo, al sistema de memoria.

EJEMPLO 8–2

```
1111 XXXX XXXX XXXX XXXX

        o bien

1111 0000 0000 0000 0000 = F0000H

        hasta

1111 1111 1111 1111 1111 = FFFFFH
```



En el ejemplo 8-2 se muestra la forma en que se determina el intervalo de direcciones de todo el decodificador. Se verá que el intervalo o alcance es la localidad F0000H hasta la FFFFFH, es decir, un bloque de 64K bytes de la memoria.

¿Cómo es posible determinar el intervalo de dirección de cada dispositivo de memoria conectado con las salidas del decodificador? También en este caso, se escribe un patrón de bits y esta vez las entradas C, B y A de dirección son no importa. En el ejemplo 8-3 se muestra la forma en que a la salida 0 del decodificador se hace baja para seleccionar la EPROM conectada con esa terminal. En este caso, se ilustra a C, y A como ceros lógicos.

EJEMPLO 8–3

```
      CBA
1111 0000 XXXX XXXX XXXX

        o bien

1111 0000 0000 0000 0000 = F0000H

        hasta

1111 0001 1111 1111 1111 = F1FFFH
```

Si se requiere el intervalo de direcciones de la EPROM conectada con la salida 1 del decodificador, se determina en la misma forma que la de la salida 0. La única diferencia es que las entradas C, B y A contienen un 001 en vez de un 000 (véase el ejemplo 8-4). Los restantes intervalos de salida de dirección se determinan en la misma forma al sustituir la dirección binaria de la terminal de salida en C, B y A.

EJEMPLO 8–4

```
      CBA
1111 001X XXXX XXXX XXXX

        o bien

1111 0010 0000 0000 0000 = F2000H

        hasta

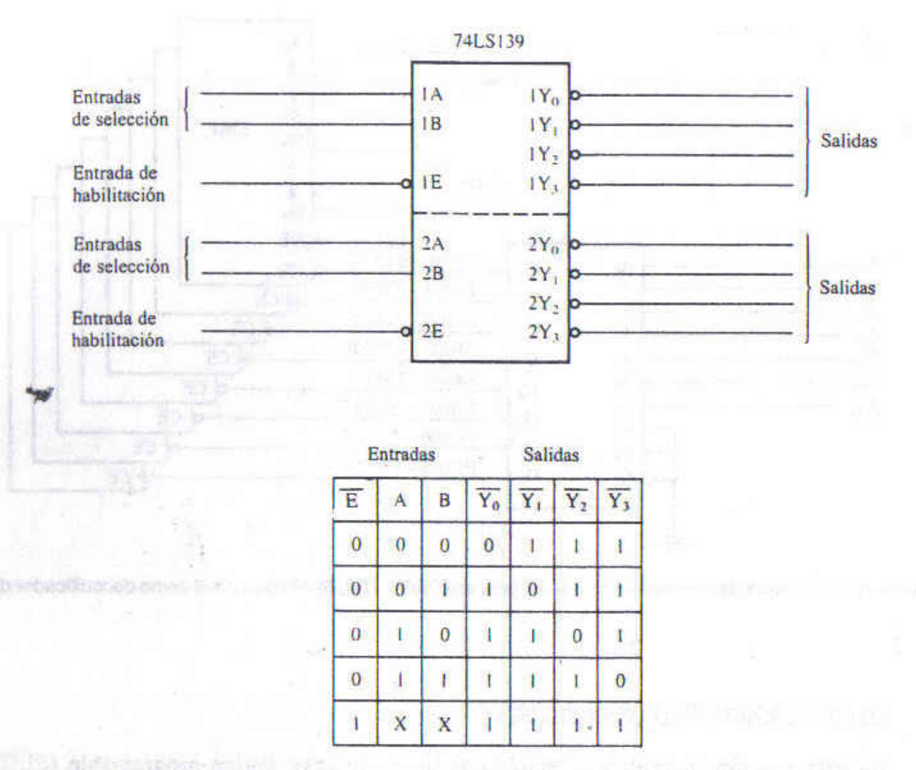
1111 0011 1111 1111 1111 = F3FFFH
```

El decodificador doble de 2 a 4 líneas (74LS139)

Otro decodificador que se aplica a veces es el de 2 a 4 líneas 74LS139. En la figura 8-15 se ilustran las conexiones en las terminales y la tabla de la verdad de este decodificador, el cual contiene dos decodificadores separados de 2 a 4 líneas, cada uno con sus propias conexiones de dirección, habilitación y salida.

Decodificador PROM de dirección

Otro decodificador común para direcciones es una PROM bipolar, que se utiliza por su mayor cantidad de conexiones de salida, con lo cual se reduce el número de otros circuitos requeridos en un decodificador de dirección de memoria en el sistema. El decodificador 74LS138 tiene seis



$\overline{E}$	A	B	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

FIGURA 8–15 Diagrama de base y tabla de verdad del decodificador doble 2 a 4 líneas 74LS139.

entradas que se utilizan para conexiones de direcciones. El decodificador PROM puede tener muchas más entradas para decodificación de direcciones.

Por ejemplo, una PROM 82S147 (512 × 8), utilizada como decodificador de dirección, tiene 10 conexiones de entrada y 8 conexiones de salida. Puede sustituir al circuito ilustrado en la figura 8-14, sin agregar la compuerta NAND de 3 entradas. Esto ahorra espacio en la tarjeta del circuito impreso y reduce el costo del sistema.

En la figura 8-16 se ilustra este decodificador de dirección con la PROM instalada; ésta es un dispositivo de memoria que se debe programar con una configuración de bits correcta, para seleccionar a los 8 dispositivos de la EPROM. La PROM en sí tiene 9 entradas de dirección que seleccionan a una de las 512 localidades internas de 8 bits en la memoria. La entrada restante ($\overline{CE}$) se debe conectar a tierra porque si las salidas de la PROM flotan a su estado de alta impedancia, entonces se podría seleccionar a una o más de las EPROM por los impulsos de ruido en el sistema.

En la tabla 8-1 se ilustra la configuración de bits binarios programada en cada localidad de la PROM a fin de seleccionar las ocho EPROM diferentes. La ventaja principal del empleo de una EPROM es que el mapa de direcciones se cambia con facilidad en el campo. Debido a que la PROM viene con todas las localidades programadas como 1 lógico, sólo se deben programar ocho de las 512 localidades. Esto le ahorra un tiempo muy valioso al fabricante.

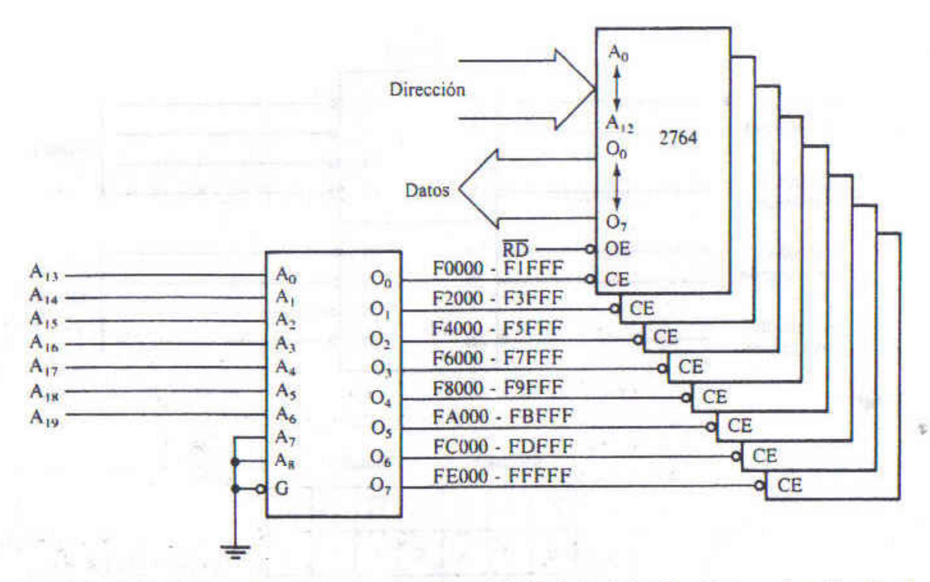


FIGURA 8–16 Un sistema de memoria en que se emplea una PROM TPB28L42 de 512 × 8 como decodificador dirección.

Decodificadores PLD programables

En esta sección se explica el empleo de un dispositivo lógico programable (PLD) como decodificador. En fechas recientes, la PAL[2] ha sustituido a los decodificadores de dirección PROM en las memorias más modernas. Hay tres dispositivos PLD que funcionan en la misma forma

[2] PAL es marca registrada de Monolithic Memories, Inc.

TABLA 8–1 Patrón de programación de la PROM 82S147 para el circuito de la figura 8-16

Entradas										*Salidas*							
OE	A8	A7	A6	A5	A4	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1	1
0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	1	1	1
0	0	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
todas las demás combinaciones de direcciones										1	1	1	1	1	1	1	1

básica pero tienen diferentes nombres: PLA (*arreglo lógico*), PAL (*arreglo de lógica programable*) y GAL[3] (*arreglo de compuertas*). Aunque estos dispositivos están en el mercado desde mediados de la década de 1970, han aparecido en fechas recientes en diseños de sistemas de memoria y digitales. El PAL y el PLA se programan al fundir un fusible como para la PROM y algunos de los GAL son dispositivos borrables igual que las EPROM. En esencia, los tres dispositivos son arreglos de elementos lógicos que se pueden programar.

Arreglos de lógica combinatoria programable. Uno de los dos tipos básicos de PAL es el arreglo de lógica combinatoria programable. Su estructura interna es un arreglo programable de circuitos lógicos combinatorios. En la figura 8-17 se ilustra la estructura interna de la PAL 16L8 que está construida con compuertas AND y OR. Este arreglo tiene 10 entradas fijas, dos salidas fijas y 6 terminales que se programan como entradas o salidas. Cada terminal de salida se genera desde una compuerta OR de 7 entradas que tiene una compuerta AND conectada con cada entrada. Las salidas de las compuertas OR pasan por un inversor de tres estados que define a cada salida como una función AND/NOR. En principio, todos los fusibles conectan a todas las conexiones verticales/horizontales que se muestran en la figura. Para la programación, se funden fusibles para conectar las diversas entradas al arreglo de compuertas OR. La función AND alambrada, se efectúa en cada conexión de entrada y produce un término producto de hasta 16 entradas. Una expresión lógica que utilice la PAL 16L8 puede tener 7 términos producto con 16 entradas juntas en las que se ejecuta la función NOR, para generar la expresión de salida. Este dispositivo es ideal como decodificador de direcciones de memoria debido a su estructura. También es ideal porque las salidas son activas en bajos.

Por fortuna, el lector no tiene que escoger los fusibles por su número para la programación. La PAL se programa con el empleo de un paquete llamado PALASM[4] (ensamblador PAL). El programa PALASM y su sintaxis son un estándar de la industria para programar los decodificadores PAL. En el ejemplo 8-5 se muestra un programa que decodifica la misma zona de la memoria que se decodifica según la figura 8-16. Se debe tener en cuenta que este programa se creó con el empleo de un editor de textos tal como el EDIT disponible con el DOS versión 5.0 de Microsoft; también se puede crear con el empleo de un editor que viene con el paquete PALASM. En algunos editores se intenta la tarea de definir las terminales, pero se cree que es más fácil utilizar EDIT y el listado que se presenta.

EJEMPLO 8–5 (página 1 de 2)

```
TITULO                  Decodificador de dirección
MODELO                  Prueba 1
REVISION                A
AUTOR                   Barry B. Brey
EMPRESA                 Symbiotic Systems
FECHA                   6/23/93
CIRCUITO INTEGRADO      Decodificador PAL 16L8
;terminales 1   2   3   4   5   6  7  8  9  10
           A19 A18 A17 A16 A13 NC NC NC NC GND

;terminales  11  12  13  14  15  16  17  18  19  20
             NC  O8  O7  O6  O5  O4  O3  O2  O1  VCC
```

[3] GAL es marca registrada de LATTICE Semiconductors, Inc.

4 PALASM es marca registrada de Monolithic Memories, Inc.

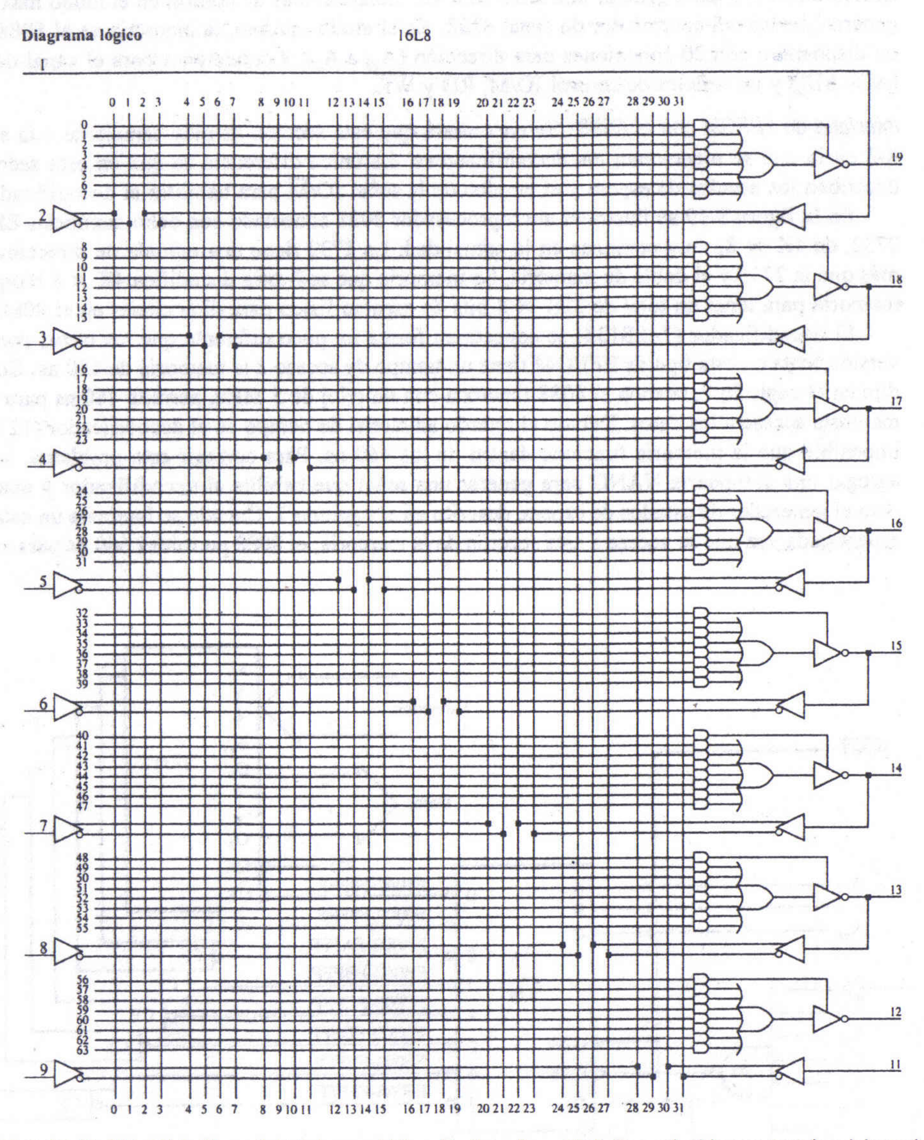


FIGURA 8–17 Un PAL 16L8. (Copyright Advanced Micro Devices, Inc., 1988. Reproducida con permiso del propietario. Todos los derechos reservados.)

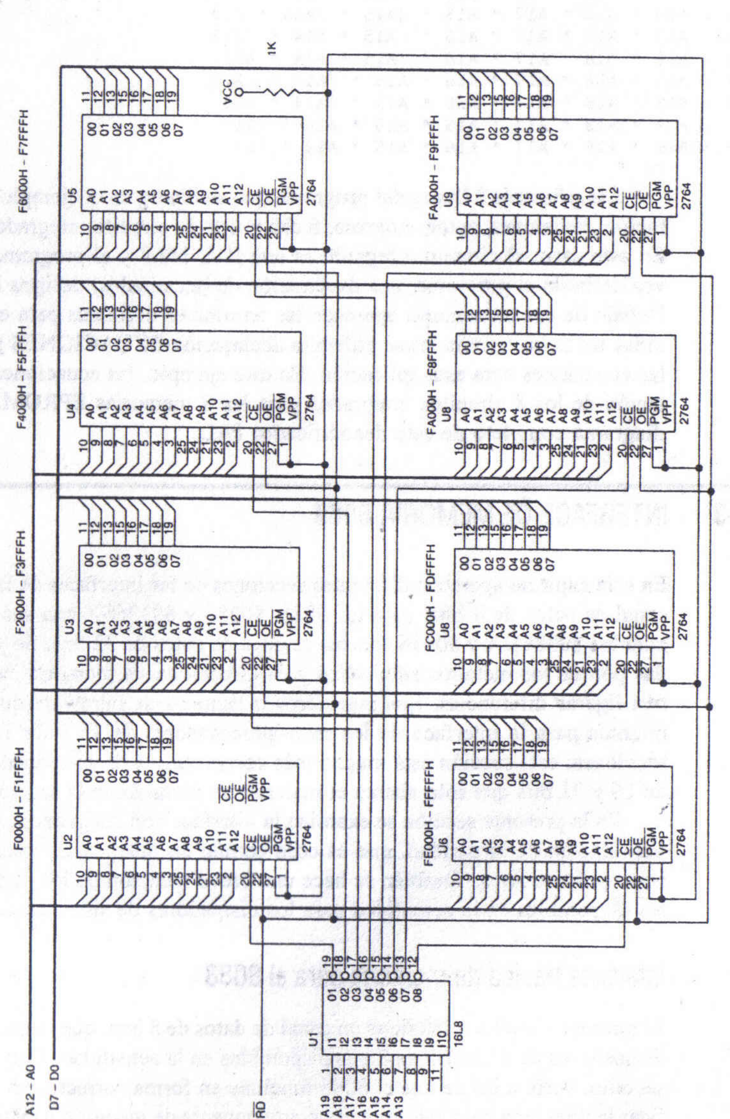


FIGURA 8–18 Un PAL 16L8 que decodifica 8 memorias 2764 (8K × 8).

EJEMPLO 8–5 (página 2 de 2)

```
ECUACIONES

/O1 = A19 * A18 * A17 * A16 * /A15 * /A14 * /A13
/O2 = A19 * A18 * A17 * A16 * /A15 * /A14 * A13
/O3 = A19 * A18 * A17 * A16 * /A15 * A14 * /A13
/O4 = A19 * A18 * A17 * A16 * /A15 * A14 * A13
/O5 = A19 * A18 * A17 * A16 * A15 * /A14 * /A13
/O6 = A19 * A18 * A17 * A16 * A15 * /A14 * A13
/O7 = A19 * A18 * A17 * A16 * A15 * A14 * /A13
/O8 = A19 * A18 * A17 * A16 * A15 * A14 * A13
```

Las primeras 8 líneas del programa que se ilustra en el ejemplo 8-5 definen el título, configuración, revisiones, autor, empresa, fecha y tipo de circuito integrado con el nombre del programa. En este caso, el circuito integrado es una PAL 16L8 y el programa se llama decodificador. Una vez definido el programa, una declaración de (terminales) designa los números de las terminales. Debajo de ese comentario aparecen las terminales definidas para esta aplicación. Unas vez definidas todas las terminales se utiliza la declaración ECUACIONES para indicar que, debajo, están las ecuaciones para esta aplicación. En este ejemplo, las ecuaciones definen las salidas de habilitación de los 8 circuitos integrados para las 8 memorias EPROM. En la figura 8-18 aparece el diagrama completo de este decodificador PAL.

8–3 INTERFACE DE MEMORIA 8088

En este capítulo aparecen diferentes secciones de las interfaces de la memoria para el 8088 con su canal de datos de 8 bits; para los 8086, 80286 y 80386SX con sus canales de datos de 16 bits y para los 80386DX y 80486 con sus canales de datos de 32 bits. Se presentan en secciones separadas porque los métodos empleados para direccionar la memoria en estos microprocesadores tienen ligeras diferencias. Los ingenieros o técnicos de hardware que deseen ampliar sus conocimientos para la interface de los microprocesadores de 16 y de 32 bits, deben leer todas estas secciones; esta sección está mucho más completa que la relacionada con la interface de memoria de 16 y 32 bits que sólo abarca el material no incluido en la sección para 8088.

En la presente sección se examina la interface con las memorias RAM y ROM y se explica la verificación de la paridad, que es cosa común en muchos sistemas de computadora basados en microprocesadores. También se hace una breve mención de los métodos para corrección de errores disponibles en la actualidad para los disipadores de sistemas de memoria.

Interface básica de memoria para el 8088

El microprocesador 8088 tiene un canal de datos de 8 bits, que lo hace ideal para conectarlo con los dispositivos de memoria de 8 bits disponibles en la actualidad. También es ideal como controlador sencillo. Pero, a fin de que el 8088 funcione en forma correcta con la memoria, ésta debe decodificar la dirección para seleccionar un componente de memoria y debe utilizar las señales de memoria $\overline{RD}$, $\overline{WR}$ y IO/$\overline{M}$ producidas por el 8088 para controlar el sistema de memoria.

La configuración para modo mínimo para el 8088 se incluye en esta sección y, en esencia, es la misma que el sistema de modo máximo para la interface de la memoria. La diferencia principal es que, en el modo máximo, IO/$\overline{M}$ se combina con $\overline{RD}$ para generar una señal $\overline{MRDC}$; IO/$\overline{M}$ se

combina con $\overline{WR}$ para generar una señal $\overline{MWTC}$. Estas señales de control en el modo máximo se generan dentro del controlador de canal 8288. En el modo mínimo, la memoria ve al 8088 como un dispositivo con 20 conexiones para dirección (A_{19} a A_0), 8 conexiones para el canal de datos (AD_7 AD_0) y las señales de control IO/$\overline{M}$, $\overline{RD}$ y $\overline{WR}$.

Interface de EPROM con el 8088. Se encontrará que esta sección es muy semejante a la sección 8-2 en la que se mencionan los decodificadores. La única diferencia es que en esta sección se describen los estados de espera y el empleo de la señal IO/$\overline{M}$ para habilitar el decodificador.

En la figura 8-19 se ilustra el microprocesador 8088 conectado con ocho memorias EPROM 2732, de 4K × 8, muy comunes en la actualidad. La 2732 tiene una entrada de dirección (A_{11}) más que la 2716 y el doble de memoria. La memoria que se ilustra decodifica 4K × 8 bloques de memoria para tener un total de 32K × 8 bits de espacio físico para direcciones en el 8088.

El decodificador (74LS138) se conecta en forma un poco diferente que los otros, porque la versión lenta de este tipo de EPROM tiene un tiempo de acceso a la memoria de 450 ns. Como se dijo en el capítulo 7, cuando el 8088 funciona con un reloj de 5 MHz, permite 460 ns para que la memoria accese a los datos. Debido al retardo adicional de tiempo en el decodificador (12 ns), es imposible que la memoria funcione dentro de los 460 ns. Para corregir este problema, se debe agregar una compuerta NAND para generar una señal que habilite al decodificador y una señal para el generador de estados de espera, descrito en el capítulo 7. Cuando se intercala un estado de espera cada vez que se accese a esta sección de la memoria, el 8088 permitirá 660 ns para accesar

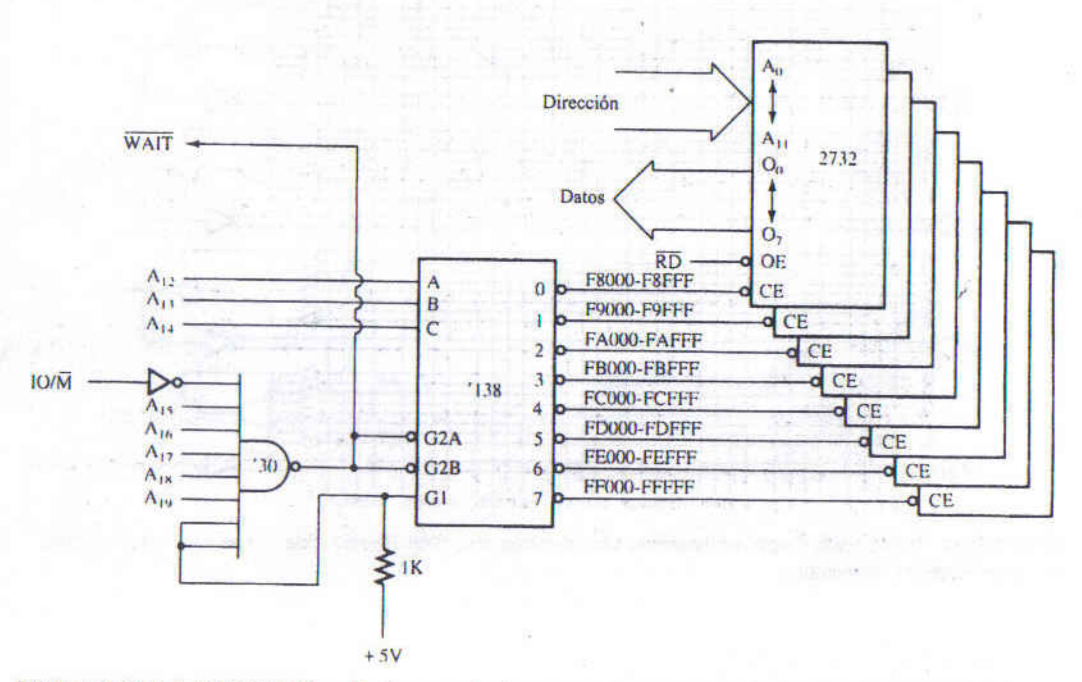


FIGURA 8–19 Ocho EPROM 2732 en interface con el microprocesador 8088. Se verá que la salida de la compuerta NAND se emplea para producir un estado de espera (WAIT) siempre que se selecciona esta sección de la memoria.

los datos de la EPROM. Recuérdese que un estado de espera adicional agrega 200 ns (1 ciclo de reloj) al tiempo de acceso. Los 660 ns son un tiempo sobrado para accesar los datos de un componente de memoria de 450 ns, incluso con los retardos introducidos por el decodificador y cualesquiera transceptor agregados en el canal de datos.

Se verá que el decodificador se selecciona para un intervalo de direcciones de memoria que empiece en la localidad F8000H y continúa hasta la FFFFFH, o sean los 64K bytes superiores de la memoria. Esta sección de la memoria es una EPROM porque la localidad FFFF0H es donde el 8088 empieza a ejecutar instrucciones después de una inicialización por hardware. A menudo, a la localidad FFFF0H se le llama de arranque en frío. El programa almacenado en esa sección de la memoria contiene una instrucción JMP en la localidad FFFF0H que brinca a la F8000H a fin de que se pueda ejecutar el resto del programa.

Interface de la RAM con el 8088. La interface para RAM es un poco más fácil que para EPROM, porque la mayor parte de los componentes de la memoria RAM no requieren estados de espera. Una sección ideal de la memoria para la RAM es la más baja, que contiene los vectores de interrupción. Los vectores de interrupción que se describen con mayor detalle en el capítulo 10, a menudo los modifican los paquetes de programación, por lo cual es de gran importancia codificar esta sección de la memoria con RAM.

En la figura 8-20 hay 16 RAM estáticas 62256, de 32K × 8, en interface con el 8088, a partir de la localidad de memoria 00000H. Esta tarjeta de circuito emplea dos decodificadores para seleccionar los componentes de las 16 diferentes memorias RAM y un tercer decodificador para seleccionar a los otros dos para las secciones correspondientes en la memoria. Hay 16 RAM de 32K que ocupan la memoria de la localidad 00000H y a la localidad 7FFFFH para tener 512K bytes de memoria.

El primer decodificador (U4) en este circuito selecciona a los otros dos decodificadores. Una dirección que empiece con 00 selecciona al decodificador U3 y una con 01 selecciona al decodificador U9. Se debe tener en cuenta que las terminales adicionales permanecen en la salida del decodificador U4 para una expansión futura. Estas permiten tener más bloques de 256K × 8 de RAM, para tener un total de 1M × 8, con sólo sumar la RAM y los decodificadores secundarios adicionales.

Además, se verá en el circuito de la figura 8-20 que todas las entradas de direcciones a esta sección de la memoria están acopladas, igual que las conexiones del canal de datos y las señales de control $\overline{RD}$ y $\overline{WR}$. El acoplamiento es importante cuando hay muchos dispositivos conectados en una sola tarjeta de circuito impreso o en un solo sistema. Supóngase que hay tres circuitos como éstos conectados en un sistema. Al no haber acoplamiento en cada circuito, la carga en los canales de direcciones, datos y control del sistema sería suficiente para impedir su funcionamiento. (La carga excesiva hace que la salida de 0 lógico se eleve a más del máximo de 0.8 volts permitidos en un sistema.) El acoplamiento se emplea, por lo general, si llega a haber adiciones a la memoria en el futuro. Si la memoria no va a crecer nunca, entonces quizá no se necesite el acoplamiento.

Paridad para detección de errores en la memoria

Debido a las memorias tan grandes disponibles en los sistemas actuales y debido a que los costos de los circuitos son muy bajos, muchos fabricantes de circuitos de memoria, han agregado la

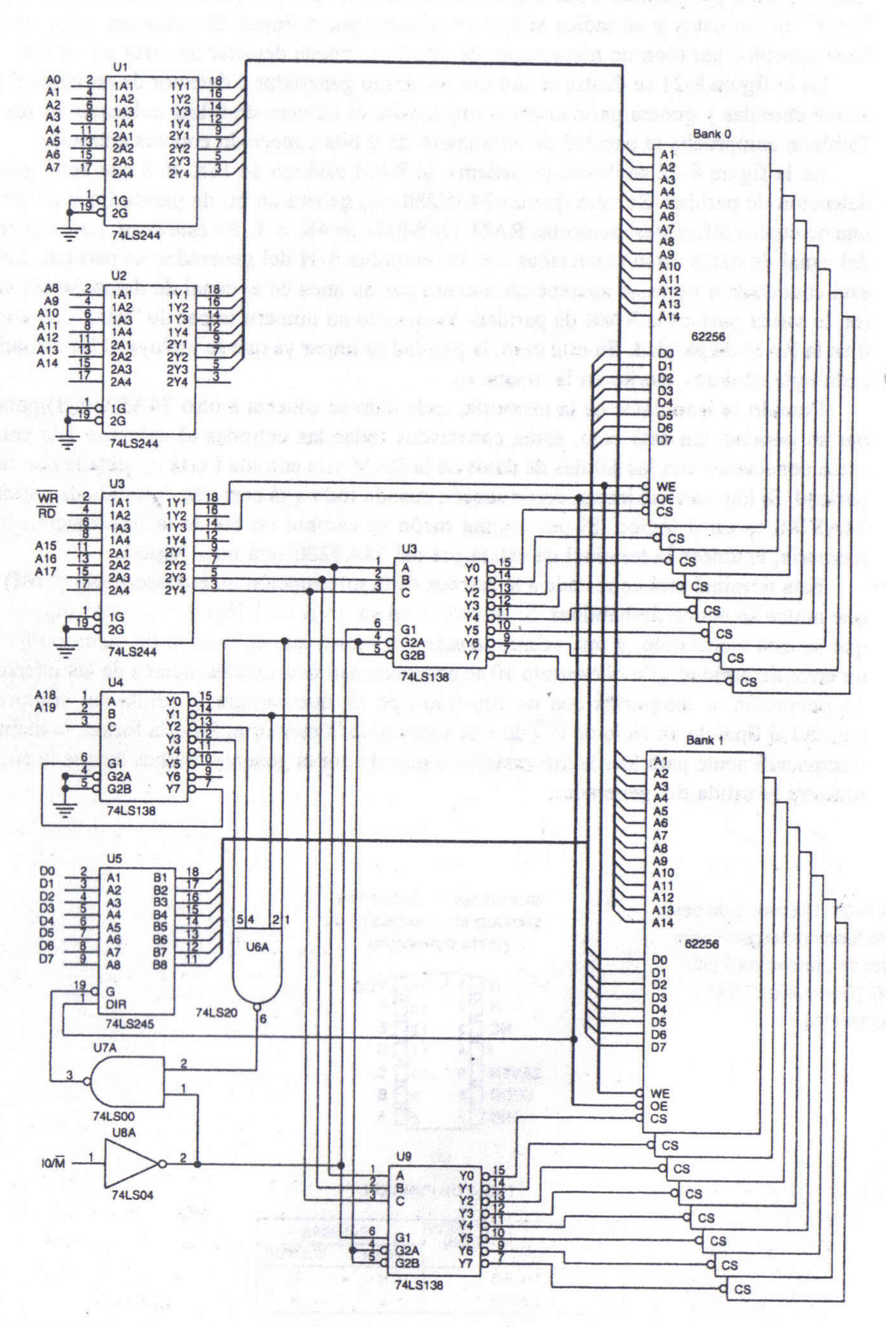


FIGURA 8–20 Un sistema de memoria estática de 512K bytes en que se emplean 16 memorias SRAM 62255.

comprobación por paridad a sus memorias RAM. Con esa comprobación se cuenta el número de "unos" en los datos y se indica si hay un número par o impar. Si todos los datos almacenados tienen paridad par (con un número par de bits 1) se puede detectar un error en un bit.

En la figura 8-21 se ilustra el circuito integrado generador y detector de paridad, el cual tiene nueve entradas y genera paridad par o impar para el número de 9 bits colocado en sus entradas. También comprueba la paridad de un número de 9 bits conectado con sus entradas.

En la figura 8-22 se ilustra un sistema de RAM estático de 16K × 8 que tiene generación y detección de paridad. Se verá que un 74AS280 (A) genera un bit de paridad que se almacena en una de cuatro diferentes memorias RAM TMS4044 de 4K × 1. En este caso, las ocho conexiones del canal de datos están conectadas con las entradas A-H del generador de paridad. La entrada I está conectada a tierra, si aparece un número par de unos en el canal de datos; se almacena un 1 (en la salida par) en la RAM de paridad. Si aparece un número impar de "unos", se almacena un 0 en la RAM de paridad. En este caso, la paridad es impar ya que se incluye el bit de paridad, para cada byte de datos escrito en la memoria.

Cuando se leen datos de la memoria, cada dato se conecta a otro 74AS280(B) para comprobar su paridad. En este caso, están conectadas todas las entradas al detector. Las entrada A-H están conectadas con las salidas de datos de la RAM y la entrada I está conectada con la RAM de paridad. Si hay paridad impar, como sucede cuando todo está correcto, la salida de paridad par del 74AS280 es un 0 lógico. Si por alguna razón se cambió un bit de la información leída de la memoria, entonces la terminal de salida par del 74AS280 será un 1 lógico.

Esta terminal está conectada a la entrada de la interrupción no enmascarable (NMI) del 8088, que nunca se puede deshabilitar. Si se coloca en su nivel de 1 lógico, se interrumpe el programa que se está ejecutando, y una subrutina especial indica que el sistema de memoria ha detectado un error de paridad. (En el capítulo 10 se proporcionan más detalles acerca de las interrupciones.) La operación se temporiza con un flip-flop tipo D, que captura la salida del comprobador de paridad al final de un ciclo de $\overline{RD}$ de esta sección de la memoria. En esta forma, la memoria tiene tiempo suficiente para leer la información y pasarla por el generador antes de que la entrada NMI muestre la salida del generador.

FIGURA 8–21 Diagrama de base y tabla de función del generador y detector de paridad de 9 bits, 74AS280. (Cortesía de Texas Instruments, Inc.)

SN54AS280 . . . PAQUETE J
SN74AS280 . . . PAQUETE N
(VISTA SUPERIOR)

G	1	14	VCC
H	2	13	F
NC	3	12	E
I	4	11	D
ΣEVEN	5	10	C
ΣODD	6	9	B
GND	7	8	A

(a)

TABLA DE FUNCION

NUMERO DE ENTRADAS HASTA 1 QUE ESTAN ALTAS	SALIDAS	
	Σ PAR	Σ IMPAR
0,2,4,6,8	H	L
1,3,5,7,9	L	H

(b)

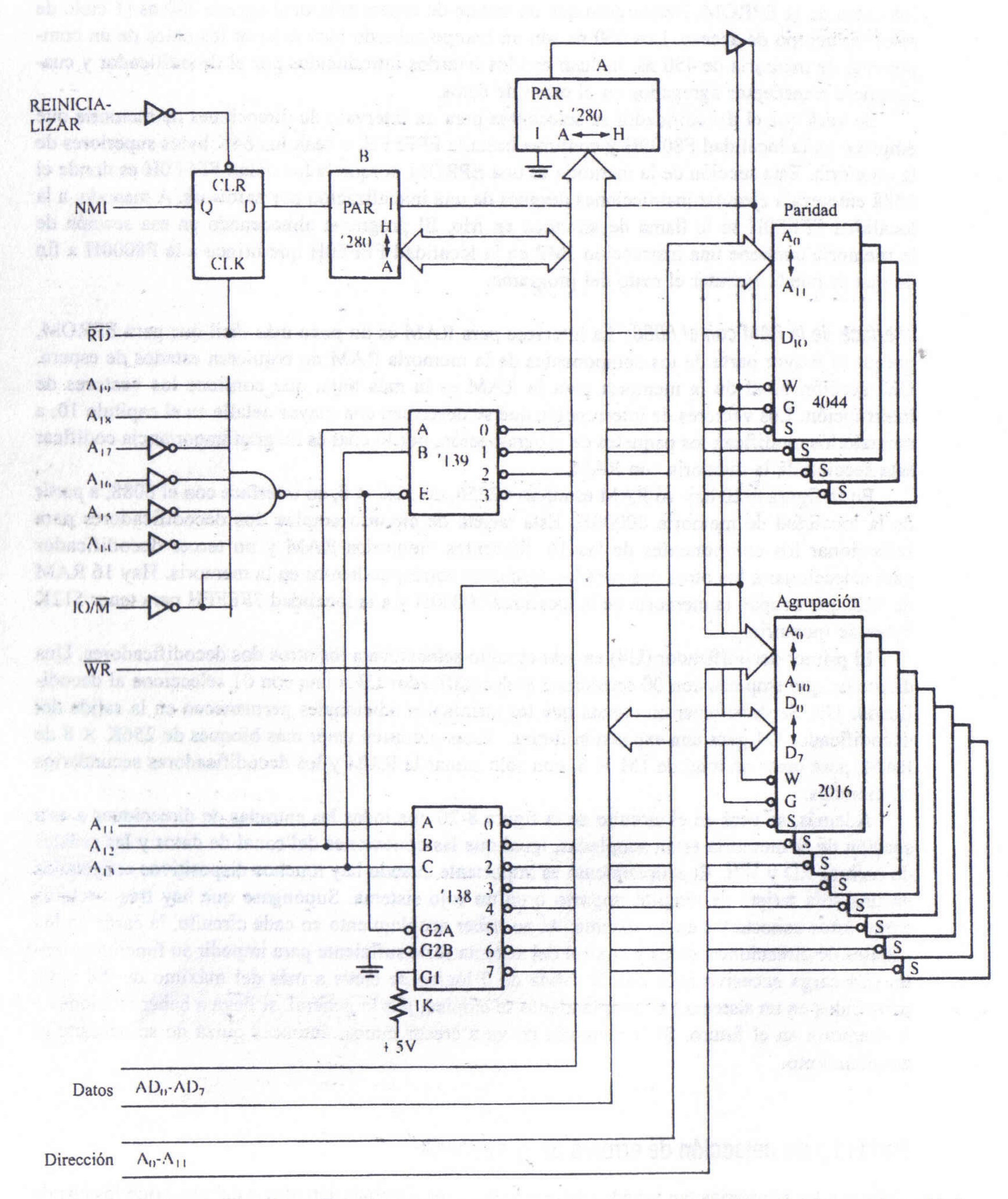


FIGURA 8–22 Un sistema de memoria de 16K × 8 que incluye un circuito de detección de error de paridad.

Corrección de errores

Los métodos para corrección de errores se han utilizado desde hace mucho tiempo, pero los fabricantes de circuitos integrados sólo han empezado hace poco a producir circuitos para corrección de errores. Si se coloca en un nivel 1 lógico, interrumpe al programa que se esté ejecutando, y una subrutina especial indica que se ha detectado un error de paridad en el sistema de memoria. (Más detalle sobre las interrupciones se da en el capítulo 10.) Uno de esos circuitos es el 74LS636, circuito para corrección y detección de errores, de 8 bits, que corrige cualquier error de un solo bit en la memoria y "levanta" una bandera para cualquier error de 2 bits.

Este circuito corrige los errores porque almacena 5 bits de paridad con cada byte de datos de la memoria. Con ello, se aumenta la cantidad requerida de memoria, pero también produce corrección automática de error de un solo bit. Si hay error en 2 o más bits, el circuito quizá no lo detecte. Por fortuna, es un caso raro y el esfuerzo adicional necesario para corregir más de un error de un solo bit es muy costoso y no vale la pena. Siempre que un componente de la memoria se daña por completo, sus bits están todos altos o todos bajos. En este caso, el circuito lo señala al procesador mediante una bandera de un error de bit múltiple.

En la figura 8-23 se ilustra el diagrama de base del 74LS636. Se verá que tiene ocho terminales de E/S de datos, cinco terminales de comprobación de bit de E/S, dos entradas de control (S_0 y S_1) y dos salidas de error: bandera de error sencillo (SEF) y bandera de error doble (DEF). Las entradas de control seleccionan el tipo de operación que se va a efectuar y se presentan en la tabla de verdad 8-2.

Cuando se detecta un error sencillo o de un solo bit, el 74LS636 efectúa un ciclo de corrección de error: coloca 01 en S_0 y en S_1, ocasiona una espera y luego hay una lectura después de la corrección del error.

En la figura 8-24 se ilustra un circuito utilizado para corregir errores de un solo bit con el 74LS636 y para interrumpir el procesador por medio de la terminal NMI para errores de doble bit. Para simplificar la descripción, sólo se ilustra una RAM de 2K × 8 y una segunda RAM de 2K × 8 para almacenar el código de comprobación de 5 bits.

La conexión de este componente a la memoria es diferente a la del ejemplo anterior. Se verá que las terminal $\overline{S}$ o $\overline{CS}$ está conectada a tierra y o un transceptor en el canal de datos controla el paso al canal del sistema. Esto es necesario si se va a accesar a los datos en la memoria antes de que la selección estroboscópica $\overline{RD}$ sea baja.

En el siguiente flanco negativo del reloj después de una señal $\overline{RD}$, el 74LS636 crea la bandera de error de un bit (SEF) para determinar si ha ocurrido un error. En tal caso, un ciclo de

TABLA 8–2 Bits de control S_0 y S_1

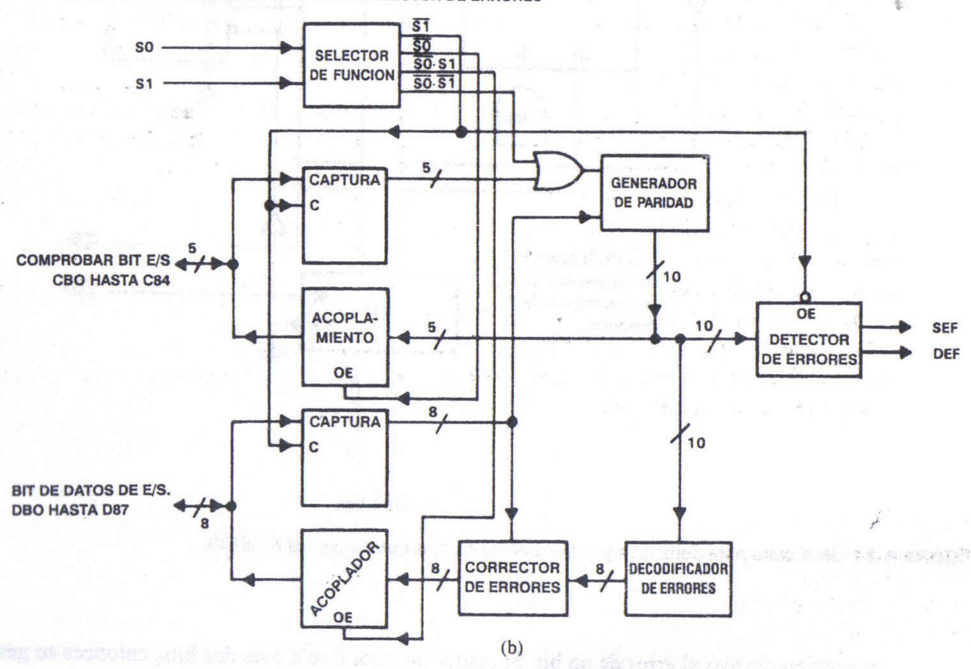
			Bandera de error	
S_0	*S_1*	*Función*	*SEF*	*DEF*
0	0	Escribir palabra comprob.	0	0
0	1	Corregir palabra de datos	*	*
1	0	Leer datos	0	0
1	1	Fijar datos	*	*

* *Nota:* Estos valores se determinan por el tipo de error.

asignación de terminales

PAQUETES J Y N			
1	DEF	11	CB4
2	DB0	12	nc
3	DB1	13	CB3
4	DB2	14	CB2
5	DB3	15	CB1
6	DB4	16	CB0
7	DB5	17	S0
8	DB6	18	S1
9	DB7	19	SEF
10	GND	20	V_{CC}

(a)



(b)

FIGURA 8–23 (a) Diagrama de base del 74LS636. (b) Diagrama de bloque del 74LS636. (Cortesía de Texas Instruments, Inc.)

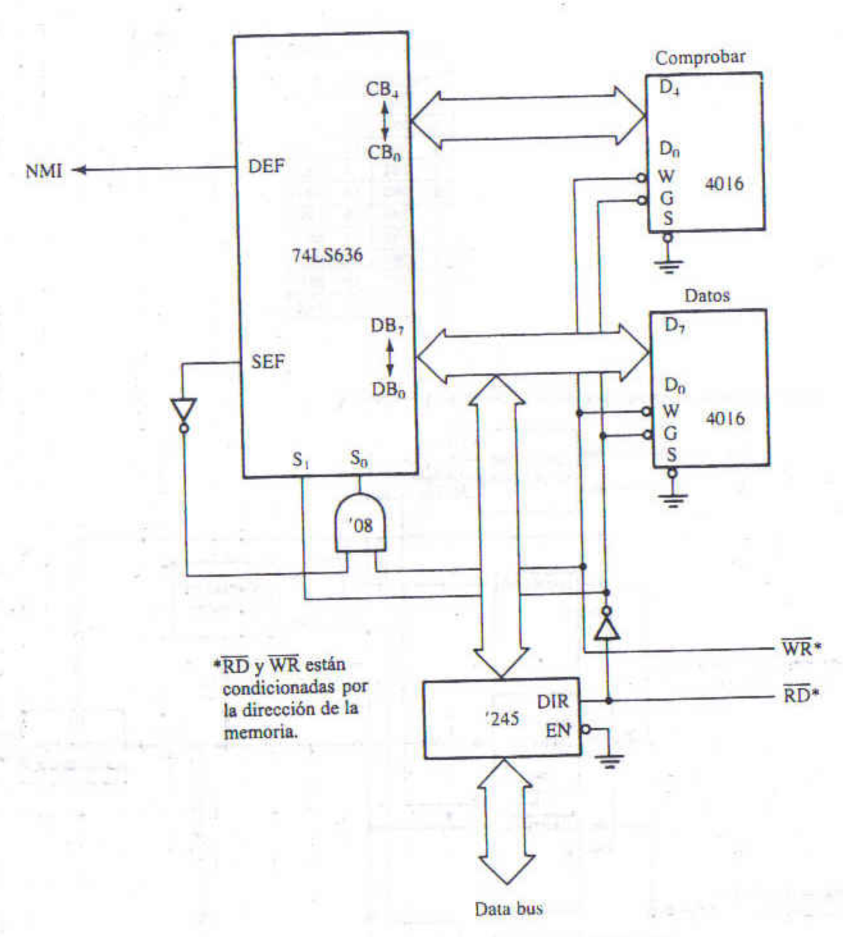


FIGURA 8–24 Un circuito para detección y corrección de errores que emplea el 74LS636.

corrección corrige el error de un bit. Si ocurre un error doble o de dos bits, entonces se genera una solicitud de interrupción por la salida de la bandera de error doble (DEF) que está conectada con la terminal NMI del microprocesador.

8–4 INTERFACE DE MEMORIA PARA LOS 8086, 80286 y 80386SX

Los microprocesadores 8086, 80286 y 80386SX difieren del 8088 en tres aspectos: (1) el canal de datos tiene 16 bits de ancho en vez de 8 bits como en el 8088, (2) la terminal $IO/\overline{M}$ el 8086 es una terminal $M/\overline{IO}$ en el 8086 y (3) hay una nueva señal de control llamada habilitación de la parte

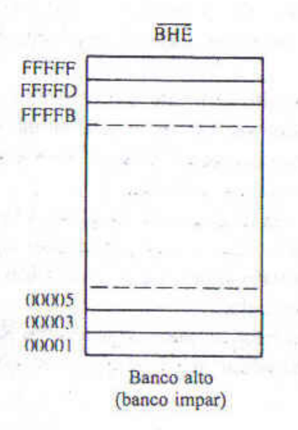
alta del canal ($\overline{BHE}$). El bit de dirección A_0 se emplea también en forma distinta. Debido a que esta sección está basada en la información dada en la sección 8-3, es indispensable leer ésta primero. Hay unas pocas diferencias más entre el 8086 y los 80286 y 80386SX; éstos, tienen un canal de dirección A_{23} hasta A_0 de 24 bits, en lugar del canal (A_{19} hasta A_0) del 8086. El 8086 tiene una señal M/$\overline{IO}$ mientras que el sistema en los 80286 y 80386SX tiene señales de control $\overline{MRDC}$ y $\overline{MWTC}$ en lugar de $\overline{RD}$ y $\overline{WR}$.

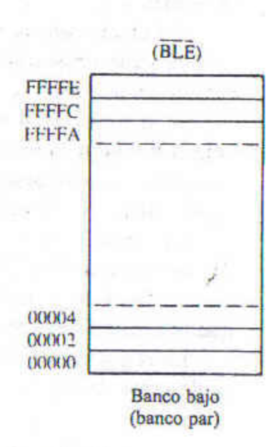
Canal de control de 16 bits

El canal de datos de los 8086, 80286 y 80386SX tiene el doble de ancho que el del 8088. Este canal más ancho plantea una serie de problemas que no se habían encontrado antes. Los 8086, 80286 y 80386SX deben poder escribir datos en cualquier localidad de 16 bits o de 8 bits. Esto significa que el canal de datos para 16 bits se debe dividir en dos secciones separadas (bancos) de 8 bits de ancho, a fin de que el microprocesador pueda escribir en una mitad (8 bits) o en ambas (16 bits). En la figura 8-25 se ilustran los dos bancos de la memoria. Un banco (banco bajo) tiene todas las localidades de memoria con número par y el otro banco (alto) contiene todas las localidades de la memoria de número impar.

Los 8086, 80286 y 80386SX emplean la señal $\overline{BHE}$ (banco alto) y el bit de dirección A_0 (banco bajo) para seleccionar uno o ambos bancos de memoria para transferir datos. La tabla 8-3 presenta los niveles de lógicos en estas dos terminales y el banco o bancos seleccionados.

FIGURA 8–25 Los bancos de memoria alto (impar) y bajo (par) de 8 bits de la memoria de los microprocesadores 8086, 80286 y 80386SX.





NOTA: A_0 está etiqueta $\overline{BLE}$ (habilitación del canal bajo) en el 80386SX.

TABLA 8–3 Selección de bancos activos de memoria con el empleo de $\overline{BHE}$ y A_0

BHE	A_0	*Función*
0	0	Ambos bancos activos (transferencia de 16 bits por D_{15} hasta D_0
0	1	Banco alto activos (transferencia de 8 bits por D_{15} hasta D_8
1	0	Bancos bajo activo (transferencia de 8 bits por D_7 hasta D_0)
1	1	Ninguno de los bancos está activo

La selección de los bancos se efectúa en dos formas: (1) se produce una señal de escritura separada para seleccionar una escritura a cada uno de los bancos de memoria o (2) se utilizan decodificadores separados para cada banco. Si se hace un estudio cuidadoso, se verá que la primera técnica es por todo el método menos costoso para la interface de memoria en los microprocesadores 8086, 80286 y 80386SX.

Decodificadores separados para cada banco. El empleo de estos decodificadores separados es, a menudo, la forma menos eficaz de decodificar direcciones de la memoria para los 8086, 80286 y 80386SX. Este método se utiliza a veces y en la mayor parte de los casos es difícil entender la razón para ello.

En la figura 8-26 se ilustran dos decodificadores 74LS138 empleados para seleccionar dos componentes de RAM de 64K para el 8086 (dirección de 20 bits). En este caso, el decodificador A tiene la terminal A0 conectada con $\overline{G2A}$ y el decodificador B tiene la señal $\overline{BHE}$ conectada con la entrada $\overline{G2A}$. Debido a que el decodificador no se activará hasta que todas sus entradas de habilitación estén activas, el decodificador A sólo se activa para una operación de 16 bits o una de 8 bits en el banco bajo y el decodificador B sólo entra en función para una operación de 15 bits o de 8 bits en el banco alto. Estos dos decodificadores y las 16 RAM de 64K bytes a las que controlan, representan todo el intervalo de 1M del sistema de memoria del 8086. Es verdad, se utilizan dos decodificadores para toda la memoria.

Se ilustra que la terminal de dirección A_0 no se conecta con la memoria, sino con el decodificador. Además, se verá que el bit A_1 del canal de direcciones está conectado a la entrada A_0 a la memoria; que A_2 está conectada con A_1, etc., La razón es que A_0 del 8086 (o del 80286 u 80386SX) ya está conectado con el decodificador A y no se necesita conectarla otra vez con la memoria. Si A_0 se conecta con la terminal de dirección A_0 de la memoria, entonces se utilizarán todas las demás localidades alternas en cada banco de la memoria. Esto significa que se desperdicia la mitad de la memoria si se conecta A_0 con A_0.

Señales estroboscópicas para escritura separadas para cada banco. La forma más efectiva para manejar la selección de banco es desarrollar una señal estroboscópica para escritura separada para cada banco de memoria. Esta técnica requiere un solo codificador para seleccionar una memoria de 16 bits de ancho. Esto, a menudo, ahorra dinero y reduce el número de componentes en un sistema.

¿Por qué no generar también señales estroboscópicas de lectura separadas para cada banco de memoria? Suele ser innecesario, porque los 8086, 80286 y 80386SX sólo leen los bytes de datos que necesitan en un momento dato, de una mitad del canal de datos. Si se presentan siempre secciones de 16 bits de datos al canal de datos durante una lectura, el microprocesador no tiene en cuenta la sección de 8 bits que no necesita, sin ningún conflicto o problemas especiales.

En la figura 8-27 se ilustra la generación de señales estroboscópica de escritura separada 8086 para la de escritura memoria. En este caso una compuerta OR 74LS32 combina A_0 con $\overline{WR}$ para la señal de selección del banco bajo ($\overline{LWR}$) y $\overline{BHE}$ con $\overline{WR}$ para la señal $\overline{HWR}$ para el banco alto. Para generar las señales estroboscópicas para los 80286 y 80386SX se utiliza la señal $\overline{MWTC}$ en vez de $\overline{WR}$.

Un sistema de memoria en que se utilizan señales estroboscópicas separadas para escritura, se construye en forma diferente al del sistema de 8 bits (8088) o el sistema en que se emplean bancos separados de la memoria. La memoria es un sistema en que se emplean señales estroboscópicas separadas para escritura, se decodifica como si fuera una memoria de 16 bits de ancho. Por ejemplo, supóngase que un sistema de memoria contendrá 64K bytes de memoria SRAM.

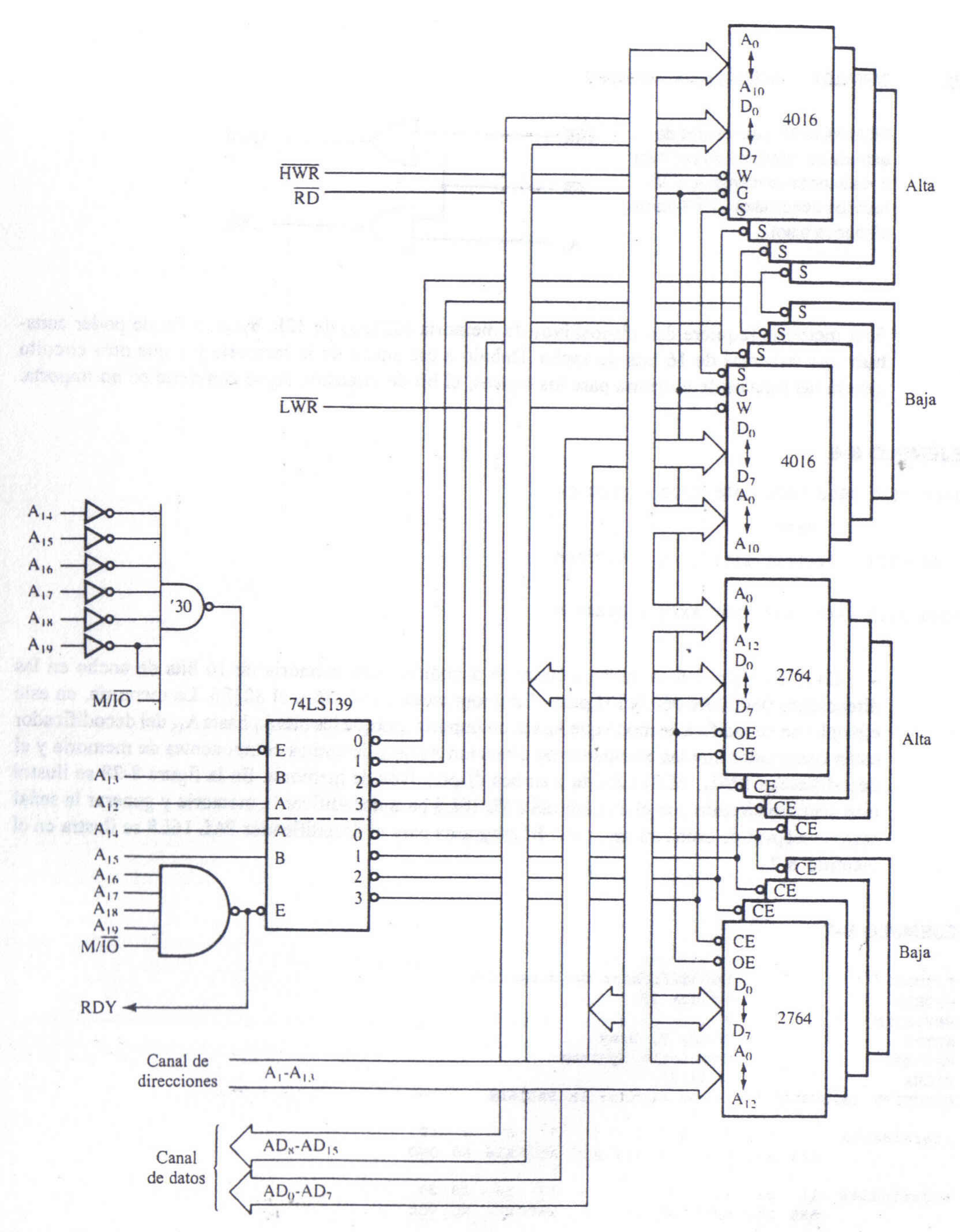


FIGURA 8–26 Una interface con la memoria de 1M byte del 8086. Se verá que no se ha intentado ilustrar $\overline{RD}$, $\overline{WR}$ ni las entradas de selección $\overline{CAS}$ y $\overline{RAS}$ a la DRAM.

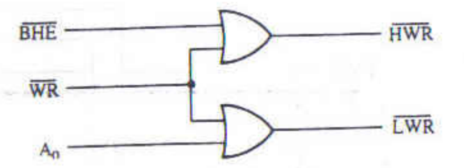


FIGURA 8–27 Las señales de entrada de selección de escritura a los bancos de memoria. $\overline{HWR}$ (escribir banco alto) y $\overline{LWR}$ (escribir banco bajo).

Esta memoria requiere dos dispositivos de memoria (62256) de 32K bytes, a fin de poder construir una memoria de 16 bits de ancho. Debido a ese ancho de la memoria y a que otro circuito genera las señales de escritura para los bancos, el bit de dirección A_0 se convierte en no importa.

EJEMPLO 8–6

```
0000 0110 0000 0000 0000 0000 = 060000H
            hasta
0000 0110 1111 1111 1111 1111 = 06FFFFH

0000 0110 XXXX XXXX XXXX XXXX = 06XXXXH
```

En el ejemplo 8-6 se muestra cómo se decodifica una memoria de 16 bits de ancho en las direcciones 060000H-06FFFFH para el microprocesador 80286 o el 80386. La memoria, en este ejemplo, se decodifica de modo que A_0 sea un importa, porque los bits A_1 hasta A_{15} del decodificador están conectados con las terminales de dirección A_0–A_{14} de ambos componentes de memoria y el decodificador (PAL 16L8) habilita a ambos dispositivos de memoria. En la figura 8-28 se ilustra este sencillo circuito con el empleo del PAL 16L8 para decodificar la memoria y generar la señal estroboscópica de escritura separada. El programa para el decodificador PAL 16L8 se ilustra en el ejemplo 8-7.

EJEMPLO 8–7

```
TITULO                 Decodificador de dirección
MODELO                 Prueba 1R
REVISION               A
AUTOR                  Barry B. Brey
EMPRESA                Symbiotic Systems
FECHA                  6/24/93
CIRCUITO INTEGRADO     Decodificador 1R PAL16L8

;terminales  1   2   3   4   5   6   7   8   9  10
             A23 A22 A21 A20 A19 A18 A17 A16 A0 GND

;terminales  11  12  13  14  15  16  17   18  19 20
             BHE SEL LWR HWR NC  NC  MWTC NC  NC VCC

ECUACIONES

/SEL = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * /A16
/LWR = /MWTC * /A0
/HWR = /MWTC * /BHE
```

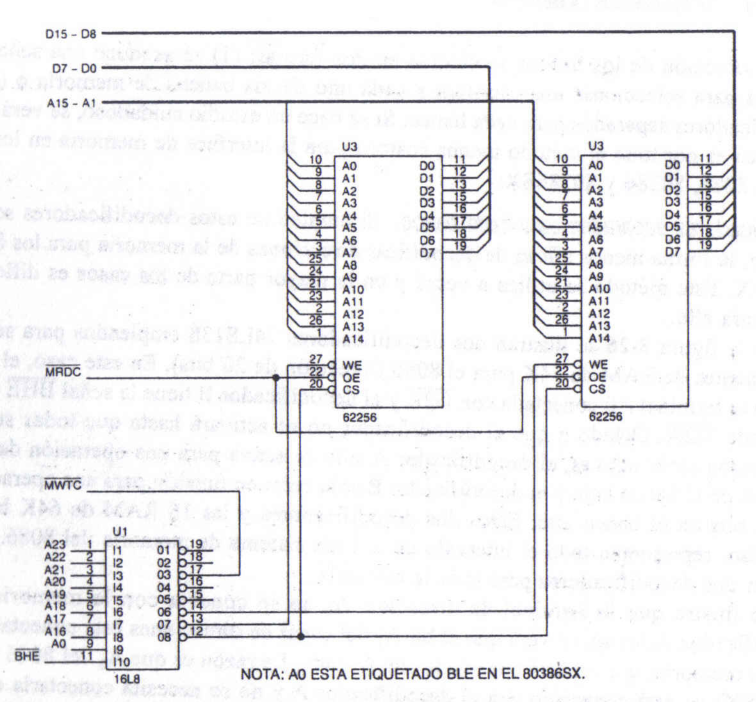


FIGURA 8–28 El decodificador de 16 bits que coloca la memoria en las direcciones 060000H hasta 06FFFFH.

En la figura 8-29 se ilustra un sistema pequeño de memoria para el microprocesador, que contiene una sección de EPROM y una sección de RAM. Hay cuatro EPROM 27128 (16K × 8) que forman una memoria de 32K × 16 bits en las F00000H hasta FFFFFH y cuatro RAM 62256 (32K × 8) que forman una memoria de 64K × 16 bits en las direcciones 00000H-1FFFFH. (Recuérdese que aunque la memoria sea de 16 bits de ancho, de todos modos se le numera en bytes.)

En este circuito se utiliza un decodificador 74LS139 doble de 2 a 4 líneas, que selecciona a la EPROM con una mitad y a la RAM con la otra mitad. Decodifica la memoria de 16 bits de ancho y no de 8 bits como antes. Se debe tener en cuenta que la señal estroboscópica $\overline{RD}$ está conectada con todas las entradas $\overline{OE}$ a la EPROM y con todas las terminales de entrada $\overline{G}$ de la RAM. Esto se hace porque si el 8086 sólo lee ocho bits de datos, la aplicación de los 8 bits restantes en el canal de datos no produce efecto en el funcionamiento del 8086.

Las señales estroboscópicas $\overline{LWR}$ y $\overline{HWR}$ están conectadas a bancos diferentes de la memoria RAM. En este caso sí importa que el microprocesador esté haciendo una escritura de 16 o de 8 bits. Si el 8086 escribe un número de 16 bits en la memoria, $\overline{LWR}$ y $\overline{HWR}$ son bajas y habilitan a las terminales $\overline{W}$ de ambos bancos de memoria. Pero, si el 8086 escribe 8 bits, entonces sólo una de las señales estroboscópicas de escritura es baja y escribe en un solo banco de memoria. También en este caso, la única ocasión en que los bancos producen una diferencia es para una operación de escritura en la memoria.

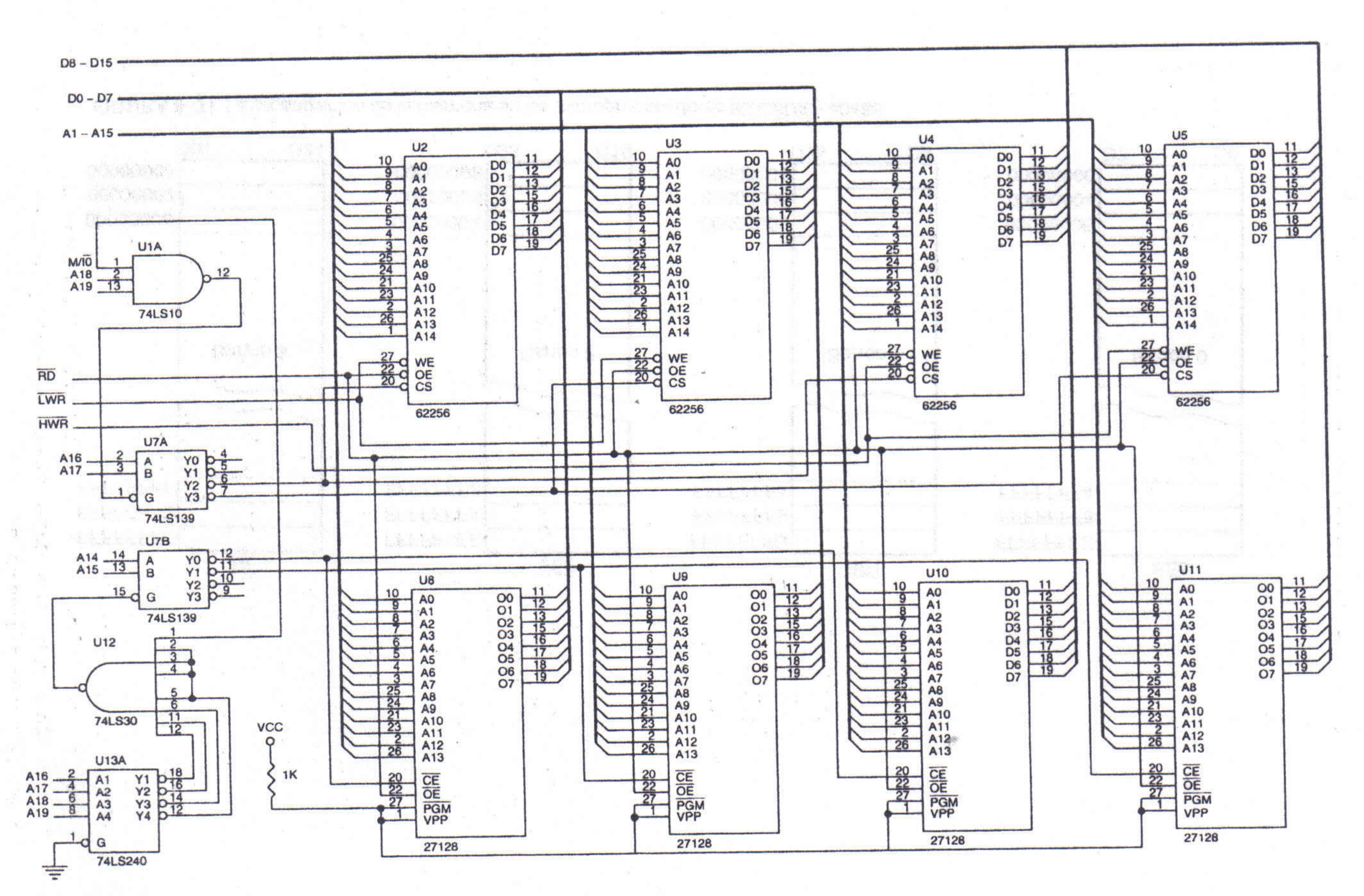


FIGURA 8–29 Un sistema de memoria para el 8086 que incluye una EPROM de 64K bytes y una SRAM de 128K bytes.

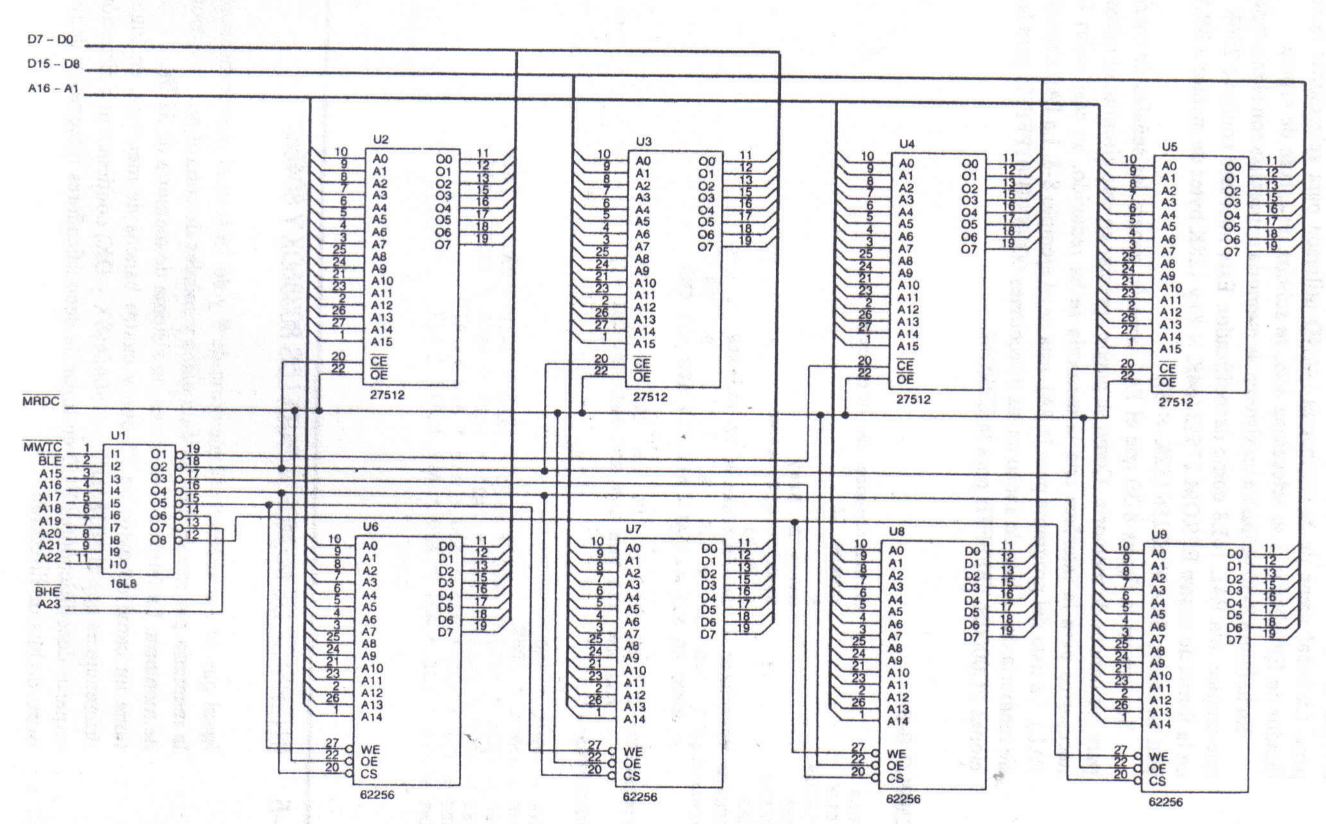


FIGURA 8–30 Un sistema de memoria del 80386SX que incluye 256K de EPROM y 128K de SRAM.

Se debe tener en cuenta que se envía una señal del decodificador de EPROM al generador de estados de espera del 8086, porque la memoria EPROM, por lo general, requiere un estado de espera. La señal viene de la compuerta NAND utilizada para seleccionar la sección del decodificador de EPROM y si se selecciona ésta, se solicita un estado de espera.

En la figura 8-30 se ilustra un sistema de memoria conectado con el microprocesador 80386SX que emplea una PAL 16L8 como decodificador. Esta interface contiene 256K bytes de EPROM, en la forma de cuatro EPROM 27512 (64K × 8) y 128K bytes de memoria SRAM, que se encuentran en cuatro SRAM 62256 (32K × 8).

Se verá en la figura 8-30 que el PAL también genera las señales de escritura $\overline{LWR}$ y $\overline{HWR}$ para los bancos de memoria. Como se puede apreciar en este circuito, el número de componentes necesarios para la interface con la memoria se ha reducido, en casi todos los casos, a uno (la PAL). La lista del programa para la PAL está en el ejemplo 8-8. La PAL decodifica las direcciones de memoria de 16 bits de ancho en las direcciones 000000H-01FFFFH para la SRAM y las direcciones FC0000H-FFFFFFH para la EPROM.

EJEMPLO 8–8

```
TITULO              Decodificador de dirección
MODELO              Prueba 1Z
REVISION            A
AUTOR               Barry B. Brey
EMPRESA             Symbiotic Systems
FECHA               6/25/93
CIRCUITO INTEGRADO  Decodificador 1Z PAL16L8
;terminales 1    2  3   4   5   6   7   8   9   10
           MWTC A0 A15 A16 A17 A18 A19 A20 A21 GND

;terminales  11  12  13  14  15  16  17  18  19  20
             A22 HWR A23 BHE LWR RB0 RB1 EB0 EB1 VCC

ECUACIONES

/LWR = /MWTC * /A0
/HWR = /MWTC * /BHE
/RB0 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * /A16
/RB1 = /A23 * /A22 * /A21 * /A20 * /A19 * /A18 * /A17 * A16
/EB0 = A23 * A22 * A21 * A20 * A19 * A18 * /A17
/EB1 = A23 * A22 * A21 * A20 * A19 * A18 * A17
```

8–5 INTERFACE DE MEMORIA PARA LOS 80386DX Y 80486

Igual que en los sistemas de memoria de 8 y de 16 bits, el microprocesador tiene interface con la memoria por medio del canal de datos y señales de control que seleccionan bancos separados de memoria. La única diferencia en un sistema de memoria de 32 bits es que el microprocesador tiene un canal de datos de 32 bits y cuatro bancos de memoria, en lugar de uno o dos. Otra diferencia es que el 80386DX y el 80486(SX y DX) contienen una dirección de 32 bits que suele requerir decodificadores PLD en lugar de decodificadores integrados, debido al considerable número de bits de dirección.

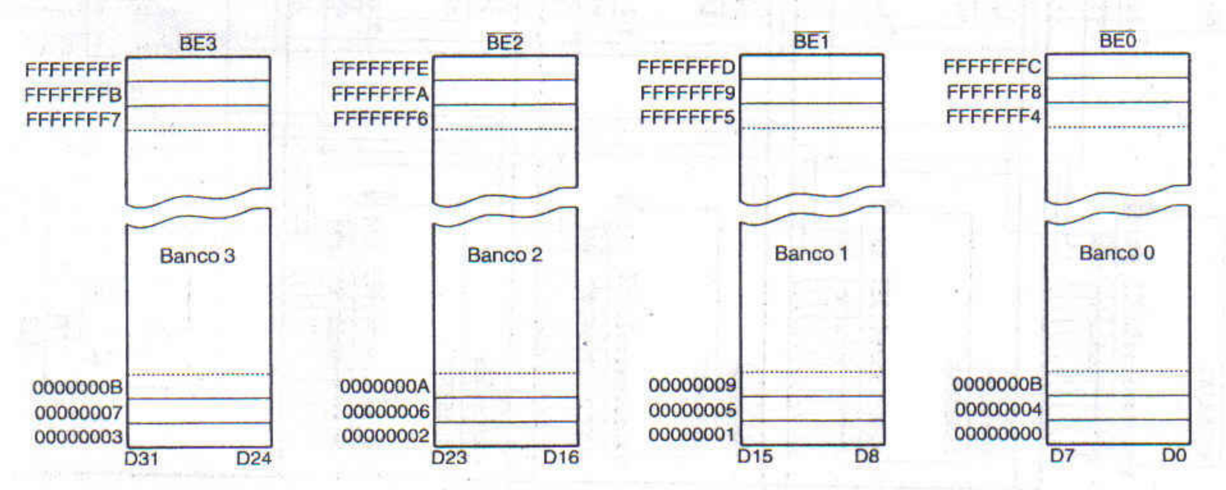


FIGURA 8–31 La organización de la memoria en los microprocesadores 80386DX y 80486.

Bancos de memoria

Los bancos de memoria para el 80386DX y el 80486, se ilustran en la figura 8-31. Se verá que estos bancos grandes de memoria contienen cuatro bancos de 8 bits y cada uno contiene hasta 1G bytes de memoria. La selección de los bancos se logra con las señales $\overline{BE3}$, $\overline{BE2}$, $\overline{BE1}$ y $\overline{BE0}$. Si se transfiere un número de 32 bits, se seleccionan los cuatro bancos; si se transfiere un número de 16 bits, se seleccionan dos bancos (por lo general $\overline{BE3}$ y $\overline{B32}$ o $\overline{BE1}$ y $\overline{BE0}$); IF se transfieren 8 bits, sólo se selecciona un banco.

Los 80386DX y 80486, igual que los 8086, 80286 y 80386SX, necesitan señales estroboscópicas de escritura separadas para cada banco de memoria. Estas señales estroboscópicas separadas para escritura, se generan como se ilustra en la figura 8-32, con el empleo de una sencilla compuerta OR u otro componente lógico.

Interface con una memoria de 32 bits

Como se puede deducir del comentario anterior, una interface de memoria para el 80386DX o el 80486, requiere que se generen cuatro señales estroboscópicas de escritura para cada banco y también que se decodifique una dirección de 32 bits.

No hay decodificadores integrados como el 74LS138 que puedan aceptar con facilidad una interface con la memoria para el 80386DX o el 80486. Se verá que los bits de dirección A0 y A1 son no importa cuando se decodifica una memoria de 32 bits de ancho. Estos bits de dirección se utilizan dentro del microprocesador para generar señales de habilitación de banco. Además, se verá que el bit A2 del canal de direcciones se conecta con la terminal A0 de dirección de la memoria.

En la figura 8-33 se ilustra un sistema de memoria de 256K × 8 para el microprocesador 80486. En esta interface se emplean ocho memorias SRAM de 32K × 8 y dos codificadores PAL 16L8. Se necesitan dos dispositivos debido al número de conexiones para dirección que hay en el 80486. Este sistema coloca la memoria SRAM en las direcciones 02000000H-0203FFFFH. El programa para los decodificadores PAL se presenta en el ejemplo 8-9.

EJEMPLO 8–9 (página 1 de 2)

```
TITULO          Decodificador de dirección
MODELO          Prueba 1U1 (PAL U1)
REVISION        A
```

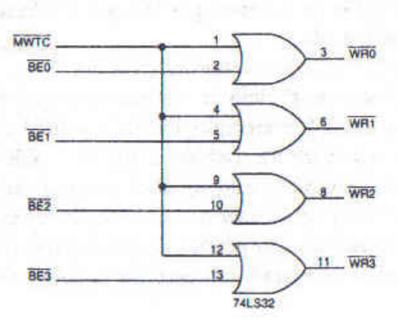
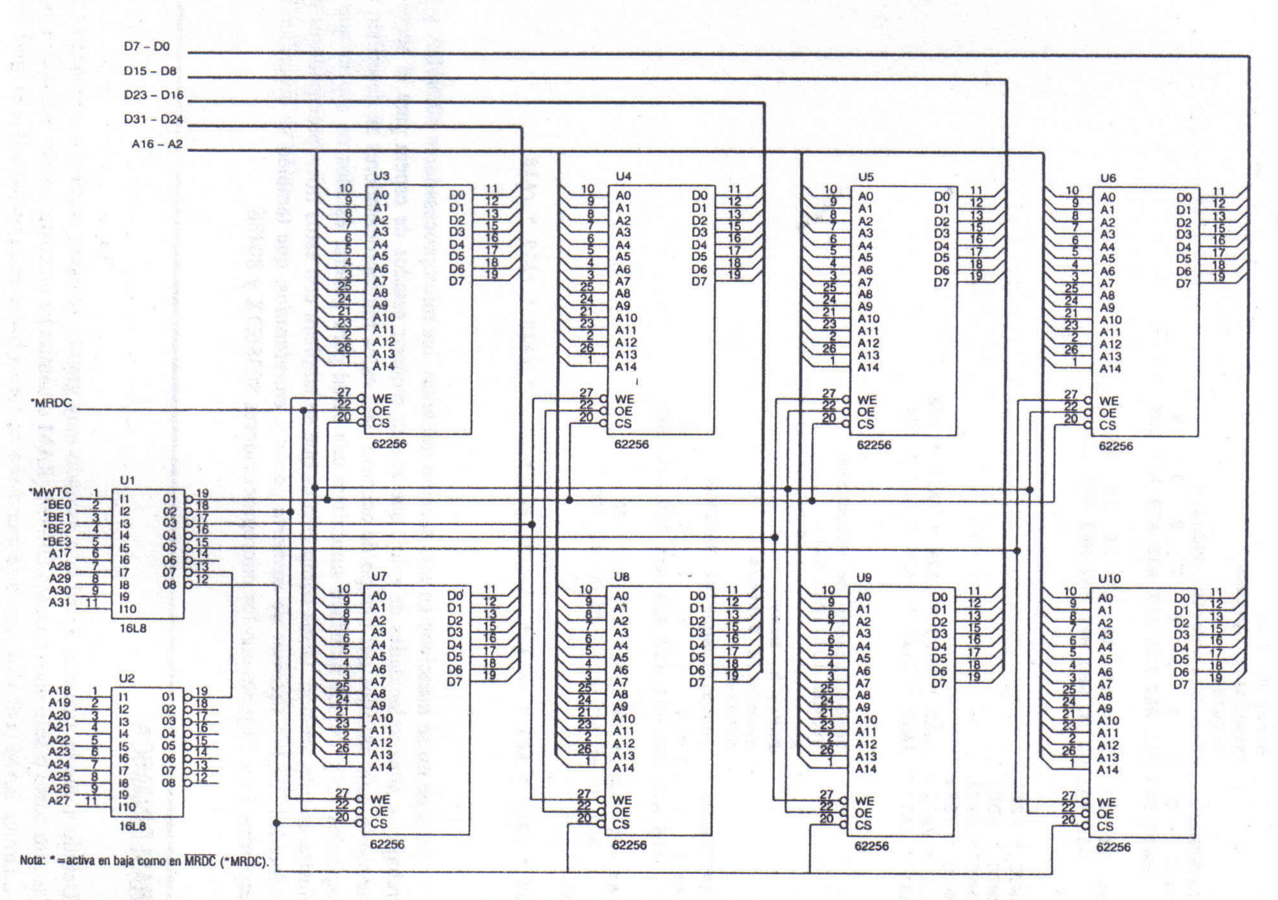


FIGURA 8–32 Señales de escritura para los bancos en los microprocesadores 80386DX y 80486.



Nota: * =activa en baja como en $\overline{\text{MRDC}}$ (*MRDC).

FIGURA 8–33 Un sistema de memoria SRAM pequeño de 256K e interface con el microprocesador 80486.

EJEMPLO 8–9 (página 2 de 2)

```
AUTOR                   Barry B. Brey
EMPRESA                 Symbiotic Systems
FECHA                   6/26/93
CIRCUITO INTEGRADO      Decodificador1U1 PAL16L8
;terminales 1    2   3   4   5   6   7   8   9   10
          MWTC BE0 BE1 BE2 BE3 A17 A28 A29 A30 GND

;terminales  11  12  13 14 15  16  17  18  19  20
             A31 RB1 U2 NC WR0 WR1 WR2 WR3 RB0 VCC
ECUACIONES

/WR0 = /MWTC * /BE0
/WR1 = /MWTC * /BE1
/WR2 = /MWTC * /BE2
/WR3 = /MWTC * /BE3
/RB0 = /A31 * /A32 * /A30 * /A29 * /A28 * /A17 * /U2
/RB1 = /A31 * /A32 * /A30 * /A29 * /A28 * A17 * /U2


TITULO                  Decodificador de dirección
MODELO                  Prueba 1U2 (PAL U2)
REVISION                A
AUTOR                   Barry B. Brey
EMPRESA                 Symbiotic Systems
FECHA                   6/26/93
CIRCUITO INTEGRADO      Decodificador1U2 PAL16L8
;terminales 1   2   3   4   5   6   7   8   9   10
           A18 A19 A20 A21 A22 A23 A24 A25 A26 GND

;terminales  11  12 13 14 15 16 17 18 19 20
             A27 U2 NC NC NC NC NC NC NC VCC
ECUACIONES

/U2 = /A27 * /A26 * A25 * /A24 * /A23 * /A22 * /A21 * /A20 * /A19 * /A18
```

Aunque no se mencione en esta sección del texto, los microprocesadores 80386DX y 80486 trabajan a altas velocidades de reloj, que suelen requerir estados de espera para el acceso a la memoria. Los cálculos del tiempo de accceso para estos microprocesadores se describen en los capítulos 13 y 14. La interface suministra una señal para generar estados de espera que no se ilustra en esta sección. Otros componentes que se emplean con estos microprocesadores de alta velocidad son los *sistemas de memoria cache* y *entrelazados*, que también se describen en los capítulos 13 y 14 junto con los microprocesadores 80386DX y 80486.

8–6 RAM DINAMICA

Debido a que la memoria RAM es a menudo muy grande, requiere muchos componentes SRAM de alto costo o sólo unos cuantos DRAM (RAM dinámica) a un costo mucho menor. La memoria dinámica RAM (DRAM), como se mencionó en forma breve en la sección 8-1, es muy compleja porque requiere multiplexión de la dirección y "refresco". Por fortuna, los fabricantes de circuitos

integrados ofrecen un controlador de RAM dinámica que incluye los multiplexores de dirección y todos los circuitos de temporización necesarios para el refresco.

En esta sección se describe la memoria DRAM con mucho más detalle que en la sección 8-1 y se da información del empleo de un controlador de DRAM en un sistema de memoria.

De nuevo con la DRAM

Como se mencionó en la sección 8-1, una DRAM retiene los datos sólo durante 2 a 4 ms y requiere la multiplexión de las entradas de dirección. Ya se han descrito los multiplexores de dirección en la sección 8-1, pero se va a describir en detalle el funcionamiento de la DRAM durante el refresco en detalle.

Como se mencionó, hay que refrescar la DRAM en forma periódica porque almacena los datos en su interior en capacitores que pierden su carga en un tiempo corto. Para refrescar una DRAM, hay que leer o escribir en forma periódica una sección de la memorial. El número de bits refrescados depende del tamaño del componente de memoria y de su organización interna.

Los ciclos de refresco se logran mediante un ciclo de lectura, de escritura o uno especial de refresco, que no lee ni escribe datos. El ciclo de refresco es completamente interno en la DRAM y se logra mientras funcionan otros componentes del sistema. Este tipo de refresco se llama refresco oculto, transparente o, a veces, robo de ciclo.

Para poder efectuar un ciclo oculto, mientras funcionan otros componentes de la memoria, un ciclo de sólo $\overline{RAS}$ introduce una señal de habilitación estroboscópica en la DRAM para seleccionar un renglón de bits que se van a refrescar. La señal $\overline{RAS}$ también hace que el renglón seleccionado se lea en el interior y se vuelva a escribir en los bits seleccionados. Con esto se vuelven a cargar los capacitores internos que almacenan los datos. Este tipo de refresco está oculto del sistema porque ocurre mientras el microprocesador lee o escribe en otras secciones de la memoria.

La organización interna de la DRAM incluye una serie de renglones y columnas. Una DRAM de 256K × 1 tiene 256 columnas, cada una con 256 bits o renglones organizados en cuatro secciones de 64K bits cada una. Siempre que se direcciona a una localidad de la memoria, la dirección de la columna selecciona una columna (o palabra interna de la memoria) de 1024 bits (una por sección de la DRAM). En la figura 8-34 se presenta la estructura interna de una DRAM de 256K × 1.

En la figura 8-35 se ilustra la temporización para un ciclo de refresco de sólo $\overline{RAS}$. La diferencia principal entre un ciclo de sólo $\overline{RAS}$ y uno de lectura o escritura, es que sólo aplica una dirección de refresco, que se suele obtener con un contador binario de 7 o de 8 bits. El tamaño del contador se determina por el tipo de DRAM que se va a refrescar. El contador de refresco se incrementa al final de cada ciclo de refresco, por lo cual se refrescan todos los renglones entre 2 y 4 ms, según sea el tipo de DRAM.

Si hay 256 renglones que se deban refrescar en 4 ms, como en una DRAM de 256K × 1, entonces hay que poner a funcionar el ciclo de refresco cada 15.6 microsegundos a fin de cumplir con la especificación del refresco. Por ejemplo, los 8086 y 8088 que trabajen a 5 MHz, necesitan 800 ns para una lectura o una escritura. Debido a que en la DRAM hay que tener un ciclo de regeneración cada 15.6 microsegundos, esto significa que por cada 19 lecturas o escrituras en la memoria, el sistema de memoria debe pasar por un ciclo de refresco o se perderán los datos que haya en la memoria. Esto representa una pérdida de 5% del tiempo de la computadora, que es un precio bajo respecto a los ahorros logrados al emplear la RAM dinámica.

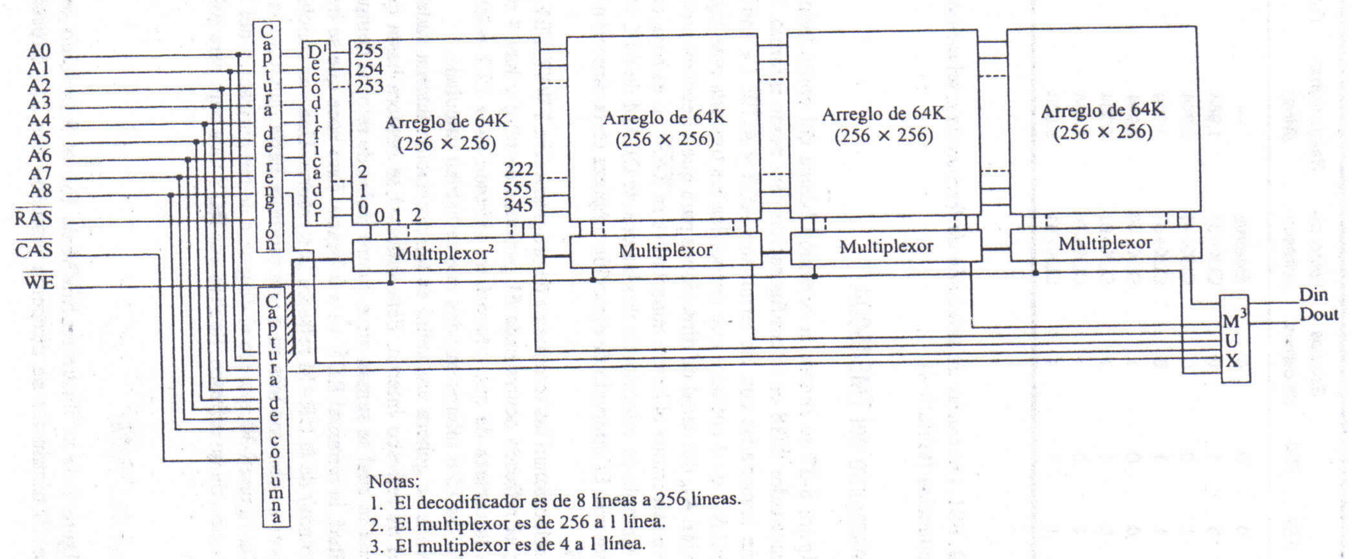


FIGURA 8–34 La estructura interna de una DRAM de 256K × 1. Se verá que cada una de las 256 palabras internas tiene 1025 bits de ancho.

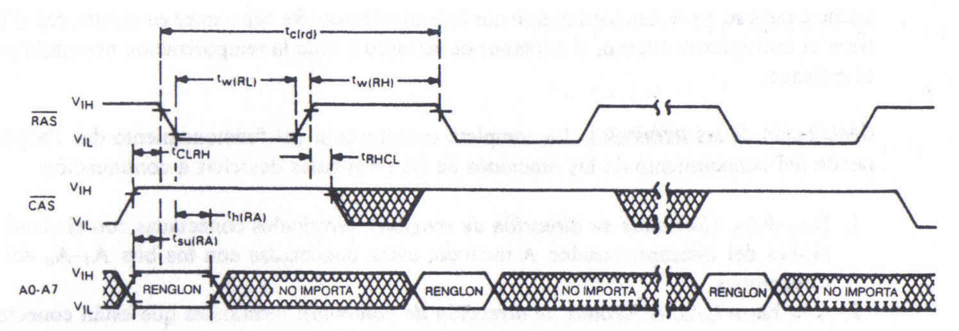


FIGURA 8–35 Diagrama de temporización del ciclo de refresco RAS para la DRAM TMS4464. (Cortesía de Texas Instruments Inc.)

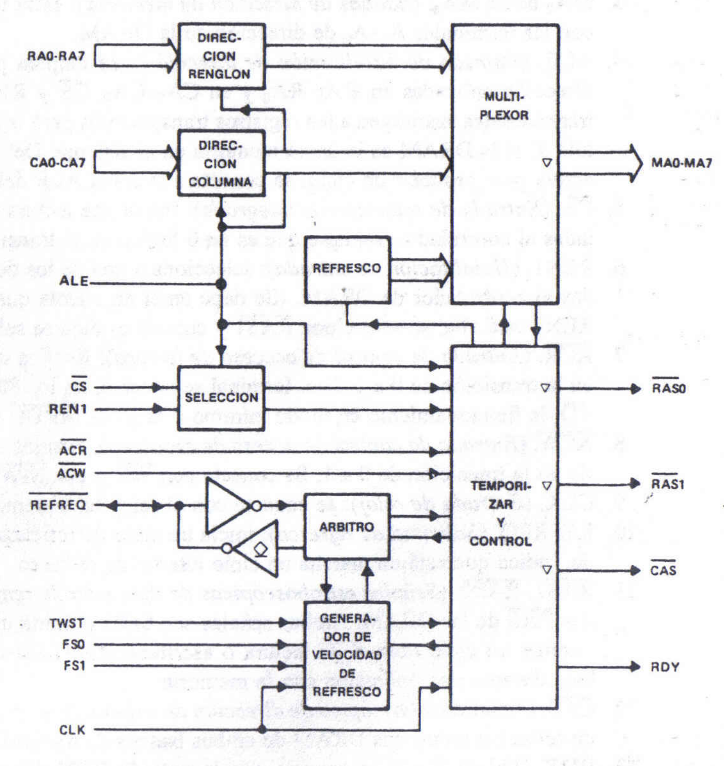


FIGURA 8–36 Diagrama de bloque del controlador TMS4500A de la DRAM. (Cortesía de Texas Instruments, Inc.)

Controladores de DRAM

De entre los muchos controladores de DRAM disponibles, en esta sección se describirá el TMS4500A (en la figura 8-36 aparece el diagrama de bloque). Igual que todos los controladores de DRAM, el TMS4500A contiene multiplexores de dirección y algún medio para solicitar un ciclo de refresco, pero al contrario de los otros, no tiene una señal especial de reloj de alta fre-

cuencia para su buen funcionamiento durante un refresco. Se debe tener en cuenta que el TMS4500 tiene el multiplexor interno, el contador de refresco y toda la temporización necesaria para lograr el refresco.

Descripción de las terminales. La completa comprensión del funcionamiento del TMS4500A depende del conocimiento de las funciones de las terminales descritas a continuación:

1. RA_7–RA_0. (*Entradas de dirección de renglón*): terminales conectadas con el canal de direcciones del microprocesador. A menudo, están conectadas con los bits A_7–A_0 del canal de direcciones.
2. CA_7 hasta CA_0. (*Entradas de dirección de columna*): terminales que están conectadas también al canal de direcciones. Si las entradas de dirección del renglón se conectan con A_7 hasta A_0, entonces estas entradas se conectan con A_{15}–A_8. El orden de estas conexiones no influye en el funcionamiento del sistema.
3. MA_7 hasta MA_0. (*Salidas de dirección de memoria*): estas terminales conectan directamente con las terminales A_7–A_0 de dirección de la DRAM.
4. ALE. (*Entrada de habilitación de dirección*): se emplea para habilitar las 16 entradas de dirección aplicadas en RA_7–RA_0 y en CA_7–CA_0, $\overline{CS}$ y REN1. Se verá que estos registros transparentes sustituyen a los registros transparentes para las direcciones descritos en el capítulo 7, si la DRAM es la única memoria en el sistema. Debido a que esta entrada debe estar activa para producir un ciclo, se conecta a la señal ALE del microprocesador.
5. $\overline{CS}$. (*Entrada de selección de integrado*): inicia una lectura o escritura en las DRAM conectadas al controlador, siempre que es un 0 lógico en la transición de 1 a 0 en ALE.
6. REN1. (*Habilitación de entrada*): selecciona a una de los dos bancos de las DRAM conectadas al controlador de DRAM. (Se debe tener en cuenta que hay dos salidas $\overline{RAS}$.) Cuando REN1 está alta, se selecciona $\overline{RAS1}$ y cuando es baja se selecciona $\overline{RAS2}$.
7. $\overline{ACR}$. (*Entrada de control de accceso de lectura*): finaliza un ciclo de lectura en la memoria en la transición de 0 a 1. Esta terminal se conecta, en los 8086/8088, con la señal de control $\overline{RD}$ de funcionamiento en modo mínimo o la señal $\overline{MRDC}$ en modo máximo.
8. $\overline{ACW}$. (*Entrada de control de acceso de escritura*): termina un ciclo de escritura en la memoria en la transición de 0 a 1. Se conecta con $\overline{WR}$ o con $\overline{MWTC}$.
9. CLK. (*Entrada de reloj*): se conecta con el reloj del sistema del microprocesador.
10. REFREQ. (*Solicitud de refresco*): inicia un ciclo de refresco o, cuando se emplea como salida, indica que está en marcha un ciclo interno de refresco.
11. $\overline{RAS1}$, $\overline{RAS0}$. (*Señales estroboscópicas de dirección de renglón*): se conectan con las entradas $\overline{RAS}$ de las DRAM. Ambas señales son bajas durante un refresco, pero sólo una es baja durante un ciclo normal de lectura o escritura. REN1 selecciona cuál terminal $\overline{RAS}$ se va baja durante una operación con la memoria.
12. $\overline{CAS}$. (*Señal estroboscópica de dirección de columna*): se conecta con todas las entradas $\overline{CAS}$ de todas las memorias DRAM de ambos bancos de memoria.
13. RDY. (*Salida lista*): se conecta con la entrada RDY del generador de reloj 8284A en un sistema con 8086 u 8088. RDY se vuelve activa cuando el controlador de DRAM efectúa un ciclo interno de refresco.
14. TWST. (*Entrada de temporización/espera*): selecciona ciclos de espera o ciertas restricciones de temporización cuando se utiliza con las entradas FS1 y FS0. Un 1 lógico en esta terminal introduce un estado de espera por cada acceso a la memoria.

TABLA 8-4 Selección de modo para el controlador de DRAM TMS4500

TWST	*FS1*	*FS0*	*Estados de espera*	*Veloc. de refresco*	*Reloj mínimo (MHz)*	*Frec. de refresco (KHz)*	*Ciclos por refresco*
0	0	0	0	External	—	REFREQ	4
0	0	1	0	CLK÷31	1.984	64-95	3
0	1	0	0	CLK÷46	2.944	64-85	3
0	1	1	0	CLK÷61	3.904	64-82	4
1	0	0	1	CLK÷46	2.944	64-85	3
1	0	1	1	CLK÷61	3.904	64-80	4
1	1	0	1	CLK÷76	4.864	64-77	4
1	1	1	1	CLK÷91	5.824	64-88	4

15. FS0, FS1. (*Entradas de selección de frecuencia*): seleccionan diversas opciones de modos y frecuencias (tabla 8-4).

Funcionamiento del TMS4500A

En la figura 8-37 se ilustra la conexión básica del controlador de DRAM con los canales del microprocesador 8088 en la configuración para modo mínimo. En este caso, las terminales RD y $\overline{WR}$ están conectadas con las terminales $\overline{ACR}$ y $\overline{ACW}$. La entrada $\overline{CS}$ se conecta con una compuerta NAND de 4 entradas que decodifica los tres bits más significativos. REN1 se conecta con la conexión A_{16} del canal de direcciones para que seleccione uno u otro banco de memoria. Si A_{16} es alta, se selecciona el banco conectado con $\overline{RAS1}$; si es baja, con el banco conectado con $\overline{RAS0}$. En este ejemplo se seleccionan dos bancos de DRAM de 64K × 8 para tener un total de 128K × 8 de memoria. El intervalo de dirección empieza en la dirección 00000H y llega hasta la localidad 1FFFFH.

Se seleccionan las terminales de programación TWST, FS1 y FS0 (0, 1 y 1) para que no haya esperas; el refresco ocurre cada 61 periodos de reloj y hay 4 ciclos de reloj para cada refresco. Esto da la certeza de que ocurrirá un refresco cada 12.2 microsegundos, que están dentro del límite de los 15.6 microsegundos que se habían calculado.

Cuando se genera una señal de temporización interna cada 61 periodos de reloj, ocurre una solicitud de refresco interna. Esta solicitud se demora hasta que concluye el ciclo de canal en curso, con lo cual se suman otros cuatro ciclos de reloj al tiempo. Cuando se hacen los honores a la solicitud, la terminal RDY va a 0 lógico. Esto hace que se inserten estados de espera mientras el controlador de la DRAM refresca la memoria. Una vez concluido el refresco, el canal está libre para llevar a cabo una lectura o una escritura hasta que haya la siguiente solicitud de refresco.

En la figura 8-38 aparece el diagrama de temporización del TMS4500A, durante la solicitud y la conclusión de un refresco. Se verá que RDY va a bajo para solicitar estados de espera durante el refresco.

Interface de DRAM

En la figura 8-39 se ilustra un SIMM de 1M × 9 (*módulo sencillo de memoria en línea*) que contiene 10 terminales de dirección provistas con una dirección multiplexada de 20 bits. Las

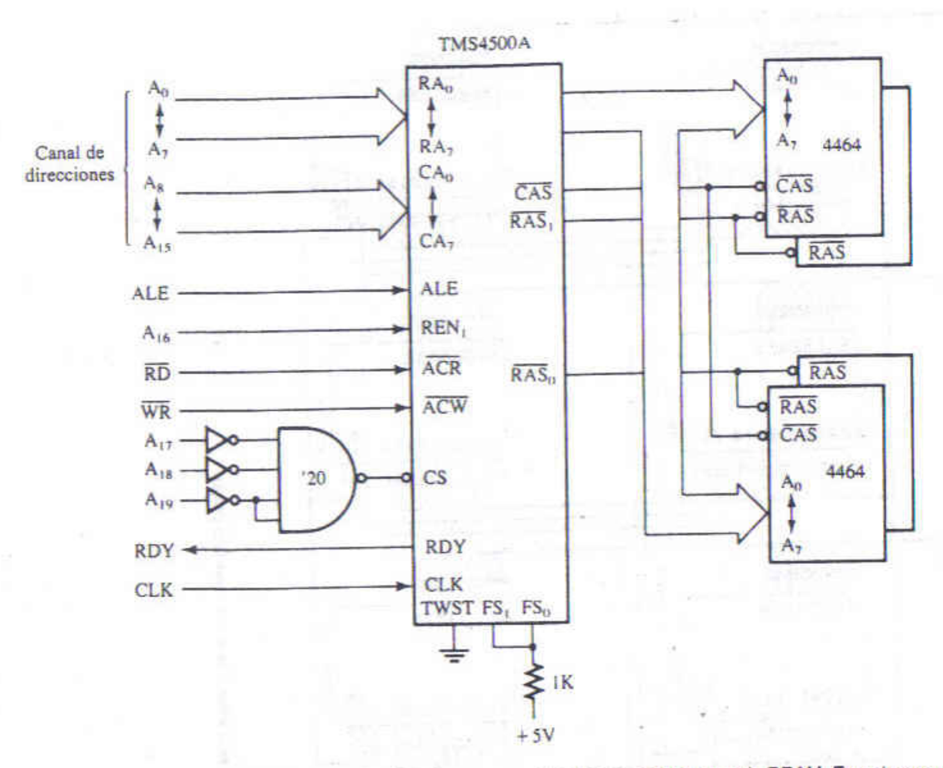


FIGURA 8–37 Controlador de DRAM TMS4500A utilizado para una interface de 128K bytes de DRAM. En este caso, cuatro DRAM TMS4464 constituyen la memoria situada en las direcciones 00000H–1FFFFH.

entradas $\overline{CAS}$ y $\overline{RAS}$ se emplean para seleccionar ($\overline{CAS}$) y las direcciones en el módulo de memoria. La entrada $\overline{CAS_9}$ selecciona el noveno bit de paridad. Este módulo también está disponible con el número, 421000C8 de 1M × 8. Las terminales del 42100C9 para el noveno bit de paridad no están conectadas en el 42100C8 y sólo hay 8 dispositivos de memoria presentes.

En la mayor parte de los más modernos sistemas de computadoras basados en un microprocesador, se utiliza el módulo de memoria SIMM. Sus tamaños usuales son 256K × 9, 1M × 9, y 4M × 9. Pronto se empezará a utilizar cada vez más el SIMM de 16M × 9.

EJEMPLO 8–10 (página 1 de 3)

```
TITLE       Address Decoder
PATTERN     Test 1U4 (PAL U4)
REVISION    A
AUTHOR      Barry B. Brey
COMPANY     Symbiotic Systems
DATE        6/27/93
CHIP        Decoder1U4 PAL16L8
```

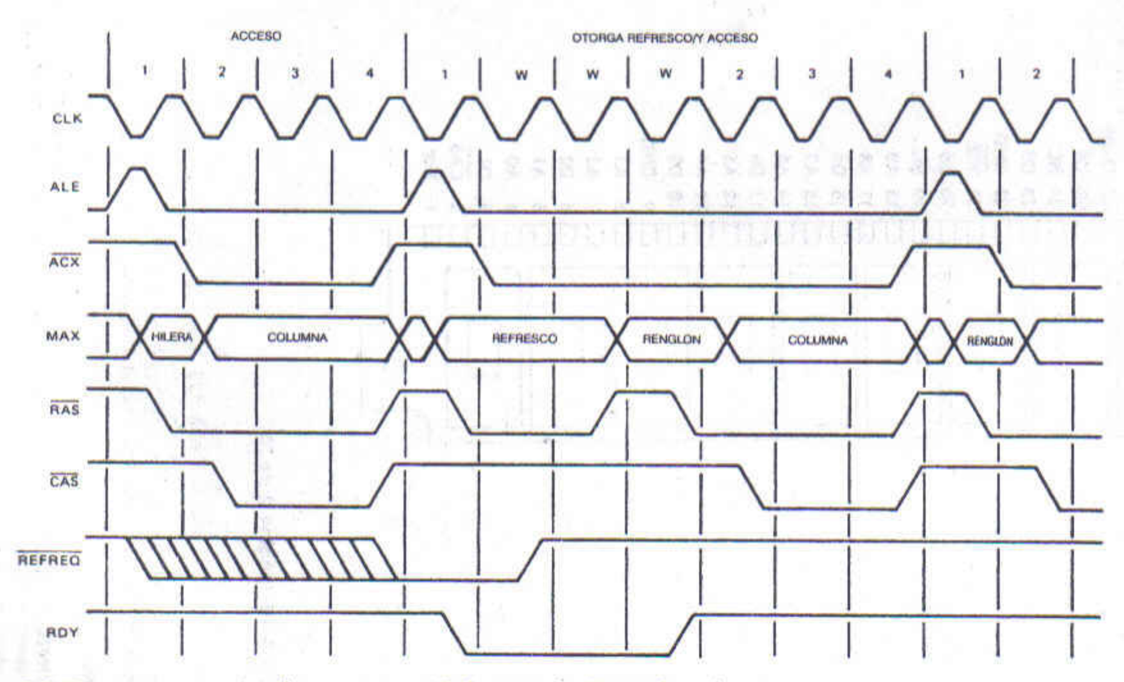
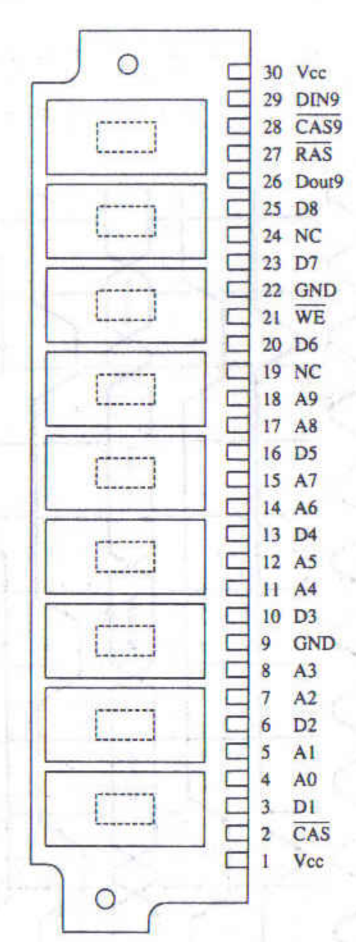


FIGURA 8–38 La temporización para el control TMS4500A de la DRAM. (Cortesía de Texas Instruments, Inc.)

FIGURA 8–39 El módulo de una sola memoria en línea (SIMM) DE 1M × 9.



EJEMPLO 8–10 (página 2 de 3)

```
;pins 1    2   3   4   5   6   7   8   9   10
      MWTC BE0 BE1 BE2 BE3 A22 A23 A24 A25 GND

;pins  11  12 13 14  15  16  17  18  19  20
       A26 NC U2 MB1 WR0 WR1 WR2 WR3 MB0 VCC

EQUATIONS

/WR0 = /MWTC * /BE0
/WR1 = /MWTC * /BE1
/WR2 = /MWTC * /BE2
/WR3 = /MWTC * /BE3
/MB0 = /A26 * /A25 * /A24 * /A23 * /A22 * /U2
/MB1 = /A26 * /A25 * /A24 * /A23 * A22 * /U2
```

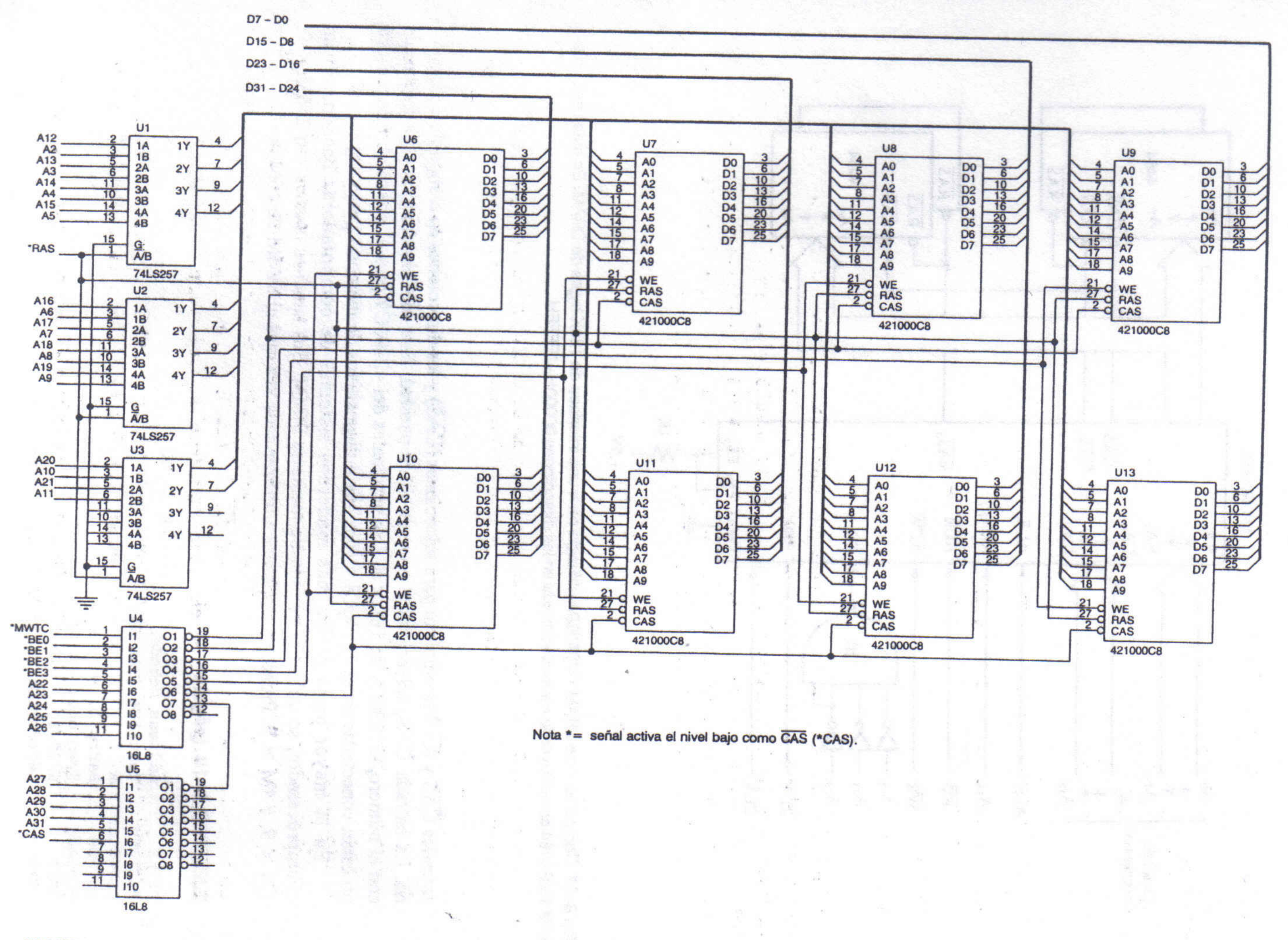


FIGURA 8–40 Interface de un sistema de memoria de DRAM con un microprocesador 80486.

EJEMPLO 8–10 (página 2 de 3)

```
TITLE      Address Decoder
PATTERN    Test 1U5 (PAL U5)
REVISION   A
AUTHOR     Barry B. Brey
COMPANY    Symbiotic Systems
DATE       6/27/93
CHIP       Decoder1U5 PAL16L8

;pins 1   2   3   4   5   6   7  8  9  10
      A27 A28 A29 A30 A31 CAS NC NC NC GND

;pins  11 12 13 14 15 16 17 18 19 20
       NC NC NC NC NC NC NC NC U2 VCC

EQUATIONS

/U2 = /A31 * /A30 * /A29 * /A28 * /A27 * /CAS
```

En la figura 8-40 se ilustra un sistema de memoria de 8M × 8 destinado al 80386DX o al 80486. No se ilustra el circuito que genera las señales $\overline{CAS}$ y $\overline{RAS}$ pero sí aparece el multiplexor de direcciones. También en este caso se emplean las PAL16L8 (véase ejemplo 8-10) para desarrollar las señales estroboscópicas de escritura para los bancos y también para decodificar la dirección en la memoria. En este caso, se decodifica el sistema de memoria para proporcionar una memoria de 8M bytes desde la dirección 00000000H hasta 007FFFFFH.

8–7 RESUMEN

1. Todos los dispositivos de memoria tienen entradas de dirección; entradas y salidas o sólo salidas de datos, una terminal para selección y una o más terminales que controlan la operación de la memoria.
2. Las conexiones de dirección en un componente de memoria se utilizan para seleccionar una de las localidades de memoria dentro del componente. Diez terminales de dirección tienen 1024 combinaciones y, por tanto, pueden direccionar 1024 localidades diferentes en la memoria.
3. Las conexiones para datos en una memoria se emplean para dar entrada a información que se almacenará en una localidad de memoria y también para recuperar información mediante lectura de localidad de la memoria. Los fabricantes designan su memoria, por ejemplo, como 4K × 4, que significa que tiene 4K (4096) localidades de memoria y 4 bits en cada una.
4. La selección de la memoria se efectúa por medio de la terminal ($\overline{CS}$) de selección de integrado en muchas RAM o con una terminal ($\overline{CE}$) en muchas memorias EPROM o ROM.
5. La función en la memoria se selecciona con la terminal ($\overline{OE}$) de habilitación de salida para leer datos y la terminal ($\overline{WE}$) para escribir datos.
6. Una memoria EPROM se programa con un programador de EPROM y se puede borrar si se la expone a luz ultravioleta. En la actualidad, las EPROM están disponibles en tamaños desde 1K × 8 hasta 128K × 8 y mayores.
7. La RAM estática (SRAM) retiene datos siempre que haya alimentación. Estos tipos de memoria están disponibles en tamaños hasta de 128K × 8.

8. La RAM dinámica (DRAM) retiene los datos, por lo general, un tiempo muy corto, de 2 a 4 ms. Esto produce problemas para el diseñador del sistema de memoria, porque hay que refrescar la DRAM en forma periódica. Las DRAM tienen también entradas de dirección multiplexadas que requieren un multiplexor externo para producir cada mitad de la dirección en el momento necesario.
9. Los decodificadores de dirección en la memoria seleccionan una EPROM o una RAM en una zona particular de la memoria. Los decodificadores de dirección que más se emplean son el 74LS138 de 3 a 8 líneas; el 74LS139 de 2 a 4 arreglos lógicos programables de selección en forma de una PROM o un PLD.
10. Los decodificadores de dirección PROM y PLD para microprocesadores como el 8088 hasta el 80486 reducen el número de circuitos integrados necesarios para formar un sistema completo de memoria que funcione.
11. La interfaz con la memoria en modo mínimo en el 8088 incluye 20 líneas de dirección, 8 líneas de datos y 3 líneas de control: $\overline{RD}$, $\overline{WR}$ y IO/$\overline{M}$. La memoria del 8088 sólo funciona en forma correcta cuando se utilizan todas estas líneas para la interface con la memoria.
12. La velocidad de acceso a la EPROM debe ser compatible con el microprocesador con el que tiene interface. Muchas de las EPROM disponibles en la actualidad, tienen un tiempo de acceso de 450 ns, que es muy lento para el 8088 de 5 MHz. Para resolver este problema se debe intercalar un estado de espera para aumentar el tiempo de acceso a la memoria a 660 ns.
13. Los comprobadores de paridad se emplean cada vez más en muchos sistemas de computadoras basados en microprocesadores. Se almacena un bit adicional con cada byte de memoria, lo cual hace que ésta sea de 9 bits en vez de 8 de ancho.
14. También hay disponibles circuitos correctores de errores para los sistemas de memoria, pero requieren el almacenamiento de muchos más bits. Si se almacena un número de 8 bits con un circuito para corrección de errores, en realidad ocupa 13 bits de memoria: 5 para el código de comprobación de error y 8 para los datos. Casi todos los circuitos integrados para corrección de error sólo tienen capacidad para corregir un error de un bit o sencillo.
15. La interface con la memoria en los 8086, 80286 y 80386SX tiene un canal de datos de 16 bits y una terminal de control M/$\overline{IO}$, mientras que el 8088 tiene un canal de datos de 8 bits y una terminal IO/$\overline{M}$. Además de estas diferencias, hay una señal adicional de control ($\overline{BHE}$), habilitar parte alta del canal.
16. La memoria de los 8086, 80386 y 80386SX está organizada en dos bancos de 8 bits: banco alto y banco bajo. El banco alto de la memoria se habilita con la señal de control $\overline{BHE}$ y el banco bajo con la señal A_0 de la dirección.
17. Dos métodos comunes para seleccionar los bancos en los sistemas basados en 8086, 80286 y 80386SX incluyen: (1) un decodificador separado para cada banco y (2) señales separadas de control $\overline{WR}$ para cada banco, con un decodificador común.
18. La interface de memoria para el 80386DX y 80486 tiene 32 bits de ancho para el ancho del canal de direcciones de 32 bits. Debido a ese ancho, la memoria está organizada en cuatro bancos cada uno de 8 bits de ancho. Las señales $\overline{BE0}$, $\overline{BE1}$, $\overline{BE2}$ y $\overline{BE3}$ para seleccionar los bancos los produce el microprocesador.
19. Los controladores de RAM dinámica se destinan a controlar los componentes de memoria DRAM. Muchos controladores de DRAM actuales incluyen multiplexores de dirección, contadores de refresco y los circuitos requeridos para efectuar el refresco periódico de la memoria DRAM.

8–8 CUESTIONARIO Y PROBLEMAS

1. ¿Qué tipos de conexiones son comunes en todos los dispositivos de memoria?
2. Enumere el número de palabras que hay en cada memoria para los siguientes números de conexiones de dirección:
   a. 8
   b. 11
   c. 12
   d. 13
3. Anote el número de datos almacenados en cada una de las siguientes memorias y el número de bits en cada dato:
   a. 2K × 4
   b. 1K × 1
   c. 4K × 8
   d. 16K × 1
   e. 64K × 4
4. ¿Cuál es la finalidad de la terminal $\overline{CS}$ o $\overline{CE}$ en un componente de memoria?
5. ¿Cuál es la finalidad de la terminal $\overline{OE}$ en un componente de memoria?
6. ¿Cual es la finalidad de la terminal $\overline{WE}$ en una RAM?
7. ¿Cuántas palabras contienen las memorias EPROM de los siguientes números?
   a. 2708
   b. 2716
   e. 2732
   d. 2764
   e. 27128
8. ¿Por qué una EPROM de 450 ns no puede funcionar directamente con un 8088 de 5 MHz?
9. ¿Qué tipo de dispositivo tiene las iniciales SRAM?
10. La memoria 4016 tiene una terminal $\overline{G}$, una terminal $\overline{S}$ y una terminal $\overline{W}$. ¿Para qué se utilizan?
11. ¿Cuánto tiempo de accceso a la memoria se requiere en la 4016 más lenta?
12. ¿Qué tipo de dispositivo tiene las iniciales DRAM?
13. La TSM4464 tiene ocho entradas de dirección, pero es una DRAM de 64. Explique cómo se puede dar una dirección de memoria de 16 bits en ocho entradas de direccionamiento.
14. ¿Cuál es la finalidad de las entradas $\overline{CAS}$ y $\overline{RAS}$ de una DRAM?
15. ¿Cuánto tiempo se requiere para refrescar la DRAM típica?
16. ¿Por qué son importantes los decodificadores de dirección de la memoria?
17. Modifique el decodificador con compuerta NAND de la figura 8-12 a fin de que pueda seleccionar la memoria para el intervalo de direcciones DF800H hasta DFFFFH.
18. Modifique el decodificador con compuerta NAND de la figura 8-12 a fin de que pueda seleccionar la memoria para el intervalo de direcciones 40000H hasta 407FFH.
19. Cuando la entrada $\overline{G1}$ está alta y $\overline{G2A}$ y $\overline{G2B}$ están bajas, ¿qué ocurre con la salida del decodificador 74LS138 de 3 a 8 líneas?
20. Modifique el circuito de la figura 8-18 para que direccione el intervalo de memoria 70000H hasta 7FFFFH.

21. Modifique el circuito de la figura 8-18 para que direccione el intervalo de memoria 40000H hasta 4FFFFH.
22. Describa el decodificador 74LS139.
23. ¿Por qué se encuentra a menudo el decodificador con dirección PROM en un sistema de memoria?
24. Vuelva a programar la PROM de la tabla 8-1 para que decodifique el intervalo 80000H hasta 8FFFFH de dirección de la memoria.
25. Vuelva a programar la PROM de la tabla 8-1 para que decodifique el intervalo 30000H hasta 3FFFFH de dirección de la memoria.
26. A las señales de control $\overline{RD}$ y $\overline{WR}$ un modo mínimo, ¿cuáles dos señales de control del 8086 las sustituyen en el modo máximo?
27. Modifique el circuito de la figura 8-18 para que seleccione la memoria de la localidad 68000H hasta 6BFFFH.
28. Modifique el circuito de la figura 8-18 para que seleccione ocho EPROM 2764 de 8K × 8 en las localidades de memoria 10000H hasta 1FFFFH.
29. Agregue otro decodificador al circuito de la figura 8-20 a fin de agregar ocho SRAM 4016 de 2K × 8 de la localidad 10000H hasta 13FFFH.
30. Vuelva a diseñar el decodificador principal de la figura 8-20 para que el direccionamiento de la memoria empiece en la dirección 80000H.
31. Explique cómo se almacena la paridad impar en un sistema de memoria y cómo se comprueba.
32. El circuito 74LS636 para detección y corrección de errores, almacena un código de comprobación con cada bytes de datos. ¿Cuántos bits se almacenan para el código de comprobación?
33. ¿Cuál es la finalidad de la terminal SEF en el 74LS636?
34. El 74LS636 corregirá _________ bits que tienen error.
35. Describa la principal diferencia entre los canales de los microprocesadores 8086 y 8088.
36. ¿Cuál es la finalidad de las terminales $\overline{BHE}$ y A_0 en el microprocesador 8086?
37. ¿Cuáles son los dos métodos que se emplean para seleccionar la memoria en el microprocesador 8086?
38. Si $\overline{BHE}$ es un 0 lógico, entonces se selecciona el banco _________ de la memoria.
39. Si A_0 es un 0 lógico, entonces se selecciona el banco _________ de la memoria.
40. ¿Por qué no se necesita producir señales estroboscópicas ($\overline{RD}$) separadas cuando se hace la interface de la memoria con el 8086?
41. Modifique el circuito de la figura 8-29 a fin de que la EPROM quede ubicada en el intervalo de memoria C0000H hasta CFFFFH y la RAM quede en el intervalo de memoria 30000H hasta 33FFFH.
42. Desarrolle una interface de memoria de 16 bits de ancho, que contenga memoria SRAM en la dirección 200000H hasta 21FFFFH para el microprocesador 80386SX.
43. Desarrolle una interface de memoria de 32 bits de ancho que contenga memoria EPROM en las direcciones FFFF0000H hasta FFFFFFFFH.
44. ¿Qué es un ciclo de sólo $\overline{RAS}$?
45. Cuando se refresca la DRAM, ¿se puede hacer mientras funcionan otras secciones de la memoria?
46. Si una DRAM de 16K × 1 requiere 2 ms para refresco y tiene 128 renglones que se van a refrescar, no debe transcurrir más tiempo que _________ antes de que se refresque otro renglón.
47. ¿Dónde se aplica la dirección de la memoria en el controlador TMS4500A para la DRAM?

48. ¿Cuál es la finalidad de la terminal REN1 en el TMS4500A?
49. ¿Qué está normalmente conectado con la terminal $\overline{ACW}$ del TMS4500A?
50. ¿Para cuál condición de funcionamiento del TMS4500A se utiliza TWST?
51. Modifique el circuito de la figura 8-37 para que seleccione el intervalo de memoria 40000H hasta 5FFFFH.
52. Modifique el programa de la PAL del ejemplo 8-10 a fin de decodificar la memoria en las direcciones 80000000H hasta 87FFFFFFH.

CAPITULO 9

Interface básica de E/S

INTRODUCCION

Un microprocesador es una gran cosa para resolver problemas, pero si no se puede comunicar con el mundo exterior, su valor es escaso. En este capítulo se presentan algunos de los métodos básicos de comunicación serial y paralela, entre los humanos o entre las máquinas y el microprocesador.

En este capítulo se presenta por primera vez la interface básica de E/S y se describe la codificación de los dispositivos de E/S. A continuación, se dan detalles de las interfaces serie y paralelo, ambas con una gran variedad de aplicaciones. Como parte de ellas se describe la conexión al microprocesador de convertidores analógico digital (AD) y digital analógico (DA), así como de motores de corriente directa (CD) y motores de paso.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Explicar el funcionamiento de las interfaces básicas de entrada y de salida.
2. Decodificar un dispositivo de E/S de 8 y de 16 bits a fin de poder utilizarlo en cualquier dirección de un puerto de E/S.
3. Definir un protocolo de comunicación ("handshake") y explicar cómo se utiliza en los dispositivos E/S.
4. Conectar y programar la interface paralela programable 8255.
5. Efectuar la conexión y programación del controlador de teclado exhibición 8279.
6. Efectuar la conexión y programación de la interface 8251A para comunicación serial.
7. Efectuar la conexión y programación del temporizador de intervalos 8254.
8. Efectuar la conexión (interface) de un convertidor analógico digital y uno digital analógico con el microprocesador.
9. Efectuar la interface de motores de CD y con el microprocesador.

9–1 INTRODUCCION A LA INTERFACE DE E/S

En esta sección se explica el funcionamiento de las instrucciones de E/S (IN, INS, OUT, OUTS). También se explica el concepto de E/S aislado (llamado a veces directo o mapeado en el espacio de E/S mapa) y E/S en el mapeado en memoria, las interfaces básicas de entrada y salida (E/S) y el protocolo de comunicación. Un conocimiento de estos temas facilitará entender la conexión y funcionamiento de los componentes programables de interface y las técnicas para E/S presentadas en el resto de este capítulo.

Instrucciones para E/S

Este conjunto de instrucciones incluye un tipo (OUT) que transfiere información a un dispositivo E/S y otro (IN) para leer información en un dispositivo de E/S. Las instrucciones (INS y OUTS que se encuentran en todas las versiones excepto los 8086 y 8088), también aparecen para transferir cadenas de datos entre la memoria y un dispositivo de E/S. En la tabla 9-1 se presentan todas las versiones de cada instrucción que se encuentran en el conjunto de instrucciones del microprocesador.

Las instrucciones IN y OUT transfieren datos entre un dispositivo de E/S y el acumulador del microprocesador (AL, AX o EAX). La dirección de E/S se almacena en el registro DX como dirección de 16 bits o en el byte (p8) que sigue inmediatamente al código de operación como dirección de E/S de 8 bits. Intel llama a la forma de 8 bits (p8) una *dirección fija* porque se almacena con la instrucción, por lo general en una ROM. La dirección de E/S de 16 bits en DX, se llama *dirección variable*, porque se almacena en un registro, que se puede variar. En las instrucciones INS y OUTS se emplean una dirección de E/S variable contenida en el registro DX.

Siempre que se transfieren datos con una instrucción IN u OUT, la dirección de E/S, llamada a menudo *número de puerto* aparece en el canal de dirección. La interface externa de E/S la decodifica en la misma forma en que decodifica una dirección en la memoria. El número (p8) del puerto fijo de 8 bits aparece en las conexiones A7 hasta A0 del canal de dirección, junto con los bits A15 hasta A8 = 0000 0000. Las direcciones por arriba de A15 son indefinidas para una instrucción de E/S. El número variable de puerto de 16 bits (DX) aparece en las terminales de dirección A15 a A0. Esto significa que las primeras 256 direcciones de puertos de E/S (00H hasta FFH) se accesan por las instrucciones de E/S fijas y variables, pero las direcciones de E/S de la 0100H hasta FFFFH sólo se accesan con la dirección variable de E/S. En muchos sistemas para tareas especificas, sólo se decodifican los 8 bits que están más a la derecha, con lo cual se reduce la cantidad de circuitos necesarios para la decodificación. En una computadora personal (PC), se decodifican los 16 bits del canal de direcciones y se emplean las localidades 00XXH hasta 03XXH para casi todos los dispositivos de E/S de la computadora.

Las instrucciones INS y OUTS direccionan al dispositivo de E/S con el empleo del registro DX, pero no transfieren datos entre el acumulador y el espacio de E/S, como ocurre con las IN y OUT. En vez de ello, estas instrucciones transfieren datos entre la memoria y un dispositivo de E/S. La dirección de memoria se apunta con ES:DI para la instrucción INS y con DS:SI para la instrucción OUTS. Al igual que con otras instrucciones para cadenas se incrementa o decrementa el contenido de los apuntadores según el estado de la bandera de dirección (DF). Las instrucciones INS y OUTS pueden llevar el prefijo REP para permitir que se transfiera más de un byte o palabra entre el espacio de E/S y la memoria.

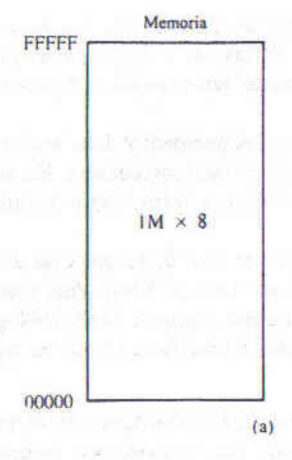
TABLA 9–1 Instrucciones para entrada y salida

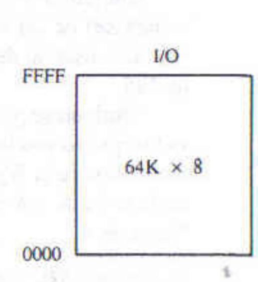
Instrucción	*Ancho de los datos*	*Comentario*
IN AL,p8	8	Leer en AL un byte de la dirección p8 de E/S
IN AL,DX	8	Leer en AL un byte de la dirección de E/S direccionada por DX
IN AX,p8	16	Leer en AX una palabra de la dirección p8 de E/S
IN AX,DX	16	Leer en AX una palabra de la dirección de E/S direccionada por DX
IN EAX,p8	32	Leer en EAX una palabra doble a partir de la dirección p8 de E/S
IN EAX,DX	32	Leer en EAX una palabra doble de la dirección de E/S direccionada por DX
INSB	8	Leer un byte de la dirección de E/S direccionada por DX y almacenarlo en la localidad direccionada por ES:DI; y después incrementar / decrementar D1 en 1
INSW	16	Leer una palabra de la dirección de E/S direccionada por DX y almacenarla en la localidad direccionada por ES:DI; luego, incrementar / decrementar DI en 2
INSD	32	Leer una palabra doble de la dirección de E/S y almacenarla en la localidad direccionada por ES:DI; luego, incrementar / decrementar DI en 4
OUT p8,AL	8	Escribir un byte de AL en la dirección p8 de E/S
OUT DX,AL	8	Escribir una palabra de AL en la dirección de E/S direccionada por DX
OUT p8,AX	16	Escribir una palabra de AX en la dirección p8 de E/S
OUT DX,AX	16	Escribir una palabra de AX en la dirección de E/S direccionada por DX
OUT p8,EAX	32	Escribir una doble palabra de EAX en dirección p8 de E/S
OUT p8,EAX	32	Escribir una doble palabra de EAX en la dirección de E/S direccionada por DX
OUTSB	8	Escribir un byte de la localidad de memoria direccionada por DS:SI en la dirección de E/S direccionada por DX; luego, incrementar / decrementar SI en 1
OUTSW	16	Escribir una palabra de la localidad de memoria direccionada por DS:SI en la dirección de E/S direccionada por DX; luego, incrementar / decrementar SI en 2
OUTSD	32	Escribir una doble palabra de la localidad de memoria direccionada por DS:SI en la dirección de E/S puesta en índice por DX; luego, incrementar / decrementar SI en 4

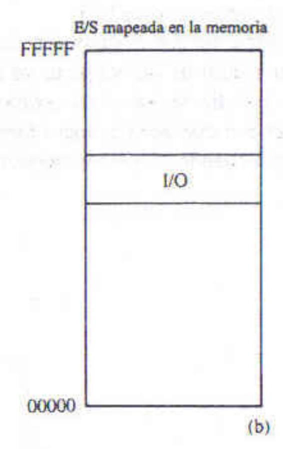
E/S aislado y mapeado en memoria

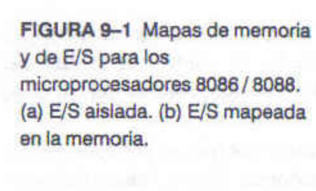
Hay dos métodos completamente distintos para hacer la interface del espacio de E/S: *aislado* y *mapeado en memoria.* En el E/S aislado, las instrucciones IN, INS, OUT y OUTS transfieren datos entre el acumulador o la memoria del microprocesador y el dispositivo de E/S. En el E/S mapeado en la memoria, cualquier instrucción que haga referencia a la memoria puede lograr la transferencia. Se utilizan y se describen ambos en este capítulo.

E/S aislado. La técnica más común para transferencia de E/S utilizada en los sistemas basados en microprocesador es de Intel, es la del E/S aislado. La palabra aislado indica la forma en que las localidades de E/S están aisladas de la memoria del sistema, en un espacio separado de direcciones de E/S. (En la figura 9-1 se ilustran el espacio de E/S aislado e incluido en la memoria para el microprocesador 8088.) Las direcciones para el espacio de E/S aislado, llamadas *puertos* están separadas de la memoria. Por ello, el usuario puede ampliar la memoria a la totalidad de su tamaño sin emplear nada de este espacio para dispositivos de E/S. Una desventaja del espacio de E/S aislado es que para accesar a los datos transferidos entre E/S y el microprocesador, se deben emplear las instrucciones IN, INS, OUT y OUTS. Se producen señales separadas para el espacio en E/S que indican una operación de lectura en E/S ($\overline{IORC}$) o una de escritura en E/S ($\overline{IOWC}$).








FIGURA 9–1 Mapas de memoria y de E/S para los microprocesadores 8086 / 8088. (a) E/S aislada. (b) E/S mapeada en la memoria.

Estas señales indican que la dirección del puerto de E/S aparece en el canal de direcciones, se emplea para seleccionar un dispositivo de E/S. En la computadora personal, se utilizan puertos de E/S aislados para controlar a los periféricos. Como regla general, se emplea una dirección de 8 bits para accesar los dispositivos ubicados en el circuito impreso del sistema, tales como la interface del temporizador y el teclado, y se emplea un puerto de 16 bits para accesar a los puertos serie y paralelo así como a los sistemas de vídeo y de unidad de disco.

E/S mapeando en memoria. En el E/S mapeando en memoria no se utilizan las instrucciones IN, INS, OUT u OUTS como en el E/S aislado. En realidad se utiliza cualquier instrucción que transfiera datos entre el microprocesador y la memoria. Un E/S incluido en la memoria se maneja como si fuera una localidad de memoria del mapa de memoria. La ventaja principal del E/S incluido en la memoria, es que para accesar el E/S se puede emplear cualquier instrucción para transferencia a la memoria. La desventaja principal es que se emplea una zona de la memoria para

mapear la E/S. Esto reduce la cantidad de memoria disponible para las demás aplicaciones. [Otra ventaja es que las señales $\overline{IORC}$ e $\overline{IOWC}$ no tienen ninguna función en un sistema de E/S incluido en la memoria, lo cual reduce el número de circuitos requeridos para la decodificación.]

Mapa de E/S de la computadora personal

En la computadora personal se utiliza una parte del mapa de E/S para funciones específicas. En la figura 9-2 se ilustra el mapa de E/S para PC. Se debe tener en cuenta que el espacio de E/S entre los puertos 0000H y 3FFH, se suele reservar para el sistema de la computadora. Los puertos de

FIGURA 9–2 Mapa de E/S de una computadora personal ilustrando muchas de las áreas fijas de E/S.

E/S en 0400H hasta FFFFH suelen estar disponibles para las aplicaciones del usuario. Se debe tener en cuenta que el coprocesador aritmético 80287 utiliza las direcciones de E/S 00F8 hasta 00FD para su comunicación. Los puertos de E/S ubicados entre 0000H y 00FFH se accesan con las instrucciones para el puerto fijo, y a los puertos ubicados más arriba de 00FFH se accesan con las instrucciones para puerto variable de E/S.

Interfaces básicas para entrada y salida

El dispositivo básico para entrada es un conjunto de registros de tres estados. El dispositivo básico para salida es un conjunto de registros transparentes. El término entrar (IN) significa transferir los datos desde el espacio de E/S hasta el microprocesador; el término salir (OUT) significa transferir los datos desde el microprocesador hasta el espacio de E/S.

Interface básica de entrada. Se utilizan registros de tres estados para construir el puerto de entrada de 8 bits mostrando en la figura 9-3. Se verá que los datos TTL externos (interruptores DIP sencillos, en este ejemplo), están conectados a la entrada del registro, cuya salida se conecta al canal de datos. Las conexiones exactas al canal de datos dependen del modelo del microprocesador. Por ejemplo, el 8088 tiene conexiones D7 - D0 en el canal de datos, mientras que el 80486 tiene D31 - D0. El circuito de la figura 9-3 permite que el microprocesador lea el contenido de los 8 interruptores que conectan con el canal de datos, cuando la señal de selección $\overline{\text{SEL}}$ se vuelve un 0 lógico.

Cuando el microprocesador ejecuta una instrucción IN, se decodifica la dirección del puerto de E/S para generar el 0 lógico en $\overline{\text{SEL}}$. Un 0 colocado en las entradas de control de salida ($\overline{\text{1G}}$) y

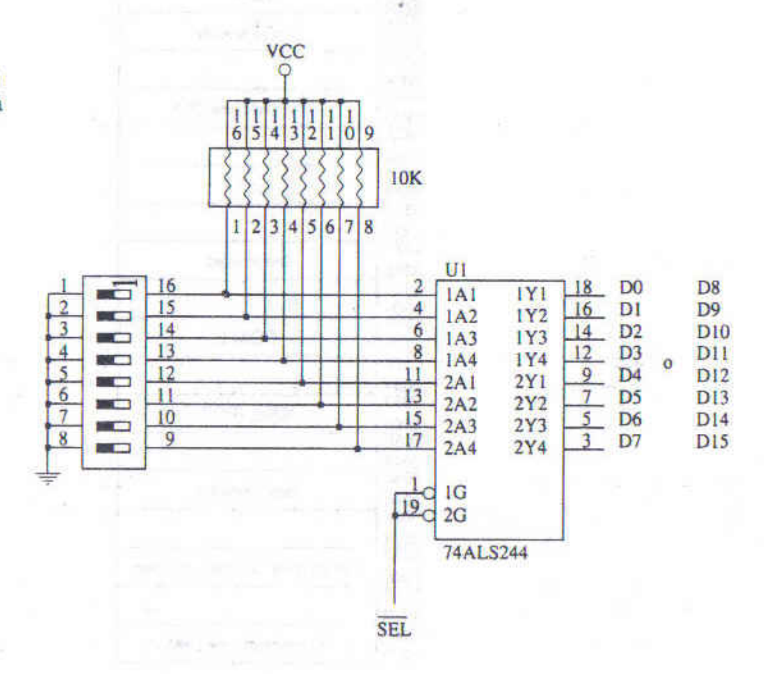


FIGURA 9–3 La interface básica de entrada ilustrando la conexión de los ocho interruptores. Se verá que la 74ALS244 es un registro de tres estados que controla la aplicación de los datos de los interruptores al canal de datos.

($\overline{2G}$) del registro 74ALS244, hace que las conexiones de entrada de datos (A) se conecten con las conexiones de salida de datos (Y). Si hay un 1 lógico en las entradas de control de salidas del 74ALS244, entra a la alta impedancia del tercer estado y desconecta los interruptores del canal de datos.

Este circuito básico de entrada no es opcional y debe actuar cada vez que se produce enlace (interface) de los datos de entrada con el microprocesador. En ocasiones, aparece como un componente discreto del circuito (figura 9-3) y, a veces, está integrado en un dispositivo programable de E/S.

También se pueden conectar datos de 16 o de 32 bits a las diversas versiones del microprocesador que no son tan comunes como los datos de 8 bits. Para conectar datos de 16 bits, se duplica el circuito de la figura 9-3 para incluir dos registros 74ALS244 que conectan los 16 bits de datos de la entrada con el canal de datos de 16 bits. Para 32 bits de datos, se amplía el circuito por un factor de 4.

La interface básica de salida. Esta interface recibe datos del microprocesador y por lo general, los retiene para algún dispositivo externo. Sus registros transparentes, como los del dispositivo de entrada, suelen estar integrados en un dispositivo de E/S.

En la figura 9-4 se ilustra la forma en que 8 sencillos diodos emisores de luz (LED) se conectan con el microprocesador por medio de un conjunto de 8 registros transparentes. En los registros se almacena el número al cual dio salida el microprocesador desde el canal de datos, para que los LED se puedan encender con cualquier número binario de 8 bits. Se necesitan estos registros para retener los datos, porque cuando el microprocesador ejecuta una instrucción OUT,

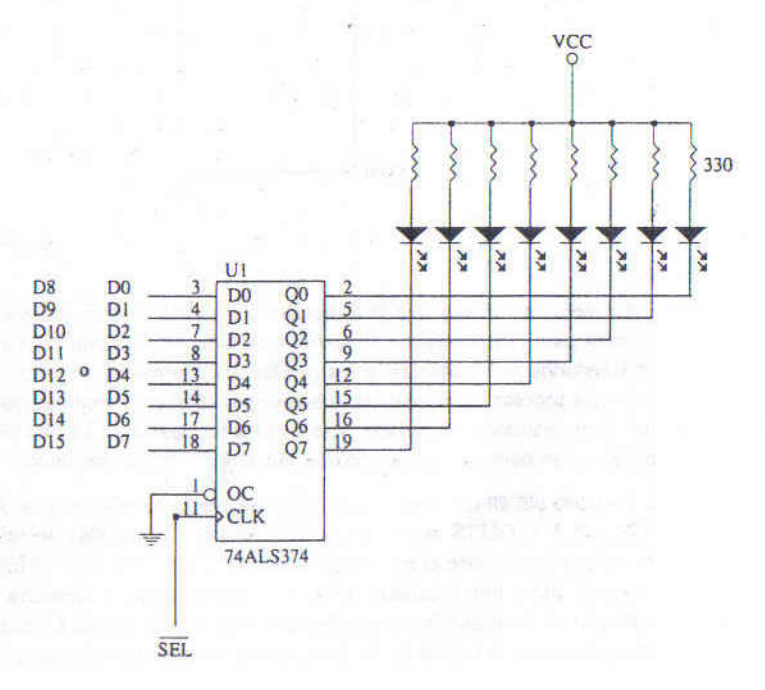


FIGURA 9–4 La interface básica de salida conectada con un grupo de LED.

los datos sólo están presentes en el canal de datos durante menos de 10 microsegundos. Al no haber estos registros, el observador nunca vería que se encendieran los LED.

Cuando se ejecuta la instrucción OUT, los datos que hay en AL, AX o EAX, se transfieren por el canal de datos, a los registros transparentes. En este caso, las entradas D de un registro transparente octal 74ALS374 se conectan al canal de datos, para "atrapar" los datos de salida; las salidas Q del registro se conectan con los LED. Cuando una salida Q se vuelve un 0 lógico, se enciende el LED. Cada vez que se ejecuta la instrucción OUT, se activa la señal $\overline{SEL}$ para el registro y "atrapa" la salida de datos del canal de datos a la salida de datos del registro. Los datos se retienen hasta que se ejecuta la siguiente instrucción OUT.

Protocolo de comunicación "handshake"

Muchos dispositivos de E/S reciben o presentan información con mucha más lentitud que el microprocesador. Otro método de control de E/S, que utiliza un protocolo determinado, sincroniza el dispositivo de E/S con el microprocesador. Un ejemplo de aparato que requiere un protocolo es una impresora que imprime 100 caracteres por segundo (CPS). Queda claro que el microprocesador puede transferir más de 100 CPS a la impresora, por lo cual hay que establecer un método para reducir la velocidad del microprocesador, para que coincida con la de la impresora.

En la figura 9-5 se ilustran las conexiones típicas de entrada y de salida en una impresora. Los datos se transfieren por medio de las terminales D7–D0 de datos. La señal BUSY (ocupado) indica que la impresora está ocupada y $\overline{STB}$ es un pulso utilizado para enviar datos para imprimir a la impresora.

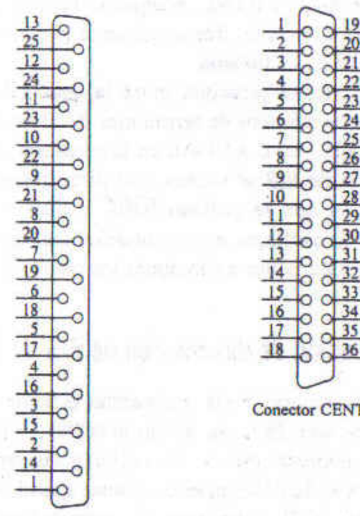
Los datos en ASCII que va a imprimir la impresora, se colocan en D7–D0 y se aplica un pulso a la $\overline{STB}$. Esta señal de habilitación estroboscópica envía los datos a la impresora para que los imprima. Tan pronto como la impresora recibe los datos, activa con un 1 lógico la terminal BUSY para indicar que está ocupada con la impresión. El microprocesador revisa o "hace una encuesta" en la terminal BUSY para decidir si la impresora está ocupada y, si lo está, el microprocesador espera; si no está ocupada, el microprocesador envía otro caracter en ASCII a la impresora. Este proceso de "interrogar" a la impresora es un protocolo de comunicación llamado reconocimiento (handshake) o "encuesta". En el ejemplo 9-1 se presenta un procedimiento sencillo para probar el contenido de la bandera BUSY de la impresora y enviar datos a ésta, si es que no está ocupada. El procedimiento Imprimir imprime el contenido en código ASCII de BL sólo si la bandera BUSY es un 0 lógico, para indicar que la impresora no está ocupada.

EJEMPLO 9–1

```
                        ;procedimiento que imprime el contenido de BL
                        ;
0000                    PRINT   PROC    FAR

0000 E4 4B                      IN      AL,BUSY         ;obtener bandera OCUPADA
0002 A8 04                      TEST    AL,BUSY_BIT     ;probar OCUPADA
0004 75 FA                      JNE     PRINT           ;si está ocupada = 1
0006 8A C3                      MOV     AL,BL           ;imprimir el carácter
0008 E6 4A                      OUT     PRINTER,AL
000A CB                         RET

000B                    PRINT   ENDP
```



DB25 Terminal No.	CENT36 Terminal No.	Función	DB25 Terminal No.	CENT36 Terminal No.	Función
1	1	$\overline{\text{Data Strobe}}$	12	12	No hay papel
2	2	Data 0 (D0)	13	13	Seleccionar
3	3	Data 1 (D1)	14	14	$\overline{\text{Afd}}$
4	4	Data 2 (D2)	15	32	$\overline{\text{Error}}$
5	5	Data 3 (D3)	16	—	$\overline{\text{RESET}}$
6	6	Data 4 (D4)	17	31	Entrada de selección
7	7	Data 5 (D5)	18—25	19—30	Tierra
8	8	Data 6 (D6)	—	17	Tierra del bastidor
9	9	Data 7 (D7)	—	16	Tierra
10	10	$\overline{\text{Ack}}$	—	33	Tierra
11	11	Ocupado			

FIGURA 9–5 El conector DB25 que se encuentra en las computadoras y en el conector para Centronics de 36 terminales que hay en las impresoras para la interface en paralelo con la impresora Centronics.

9–2 DECODIFICACION DE DIRECCIONES DE PUERTOS DE E/S

La decodificación de las direcciones en los puertos de E/S es muy similar a la de direcciones en la memoria, en especial para los E/S mapeados en la memoria. No se va a comentar la decodificación de E/S mapeado en memoria, porque es casi igual que para la memoria, excepto que no se utilizan

las señales $\overline{IORC}$ y $\overline{IOWC}$, porque no hay instrucciones IN o OUT. La decisión de mapear E/S en memoria se determina frecuentemente por el tamaño de la memoria y por la posición de dispositivos de E/S en el sistema.

La diferencia principal entre la decodificación de la memoria y la decodificación de E/S aisladas, es el número de terminales de dirección conectadas en el decodificador. Se decodifican A32–A0, A23–A0 o A19–A0 en la memoria y A15–A0 para E/S aislada. En ocasiones, si en los dispositivos de E/S se utiliza sólo direccionamiento fijo, nada más se decodifican A7–A0. Otra diferencia es que se utilizan $\overline{IORC}$ e $\overline{IOWC}$ para habilitar a los dispositivos de E/S para una lectura o una escritura. en los modelos anteriores de microprocesadores, se utilizaban IO/$\overline{M}$ = 1 y $\overline{RD}$ o $\overline{WR}$ para poner a funcionar los dispositivos de E/S.

Decodificación de direcciones de E/S de 8 bits

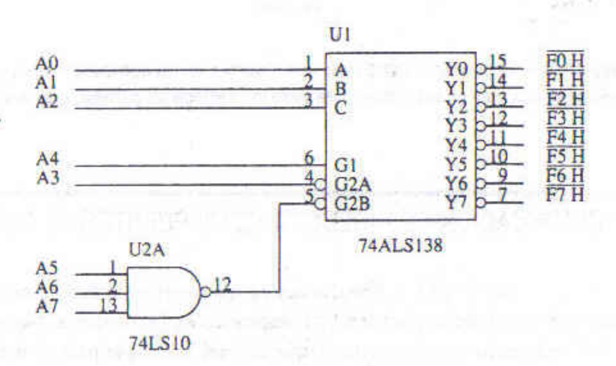
Como se mencionó, en la instrucción E/S fija se emplea una dirección de puerto de E/S de 8 bits que aparece en A15 hasta A0 como 0000H–00FFH. Si un sistema tiene menos de 256 dispositivos de E/S, a menudo sólo se decodifican las terminales de las direcciones A7–A0 para un puerto E/S de 8 bits. Se debe tener en cuenta que el registro DX también puede direccionar a los puertos de E/S 00H–FFH. Asimismo, si la dirección que se decodifica es de 8 bits, entonces nunca se podrán incluir dispositivos E/S que tengan una dirección de 16 bits.

En la figura 9-6 se ilustra un decodificador 74ALS138 para los puertos F0H hasta F7H de E/S de 8 bits. (Se supone que en el sistema sólo habrá puertos E/S 00H hasta FFH para este decodificador.) Este decodificador es muy similar al de dirección de memoria, excepto que en sus entradas sólo se conectan los bits de direcciones A7–A0. En la figura 9-7 se ilustra la versión PAL de este decodificador. Se debe tener en cuenta que es un mejor circuito decodificador porque el número de circuitos integrados se ha reducido a un solo componente: el PAL. El programa para el PAL aparece en el ejemplo 9-2.

EJEMPLO 9–2 (página 1 de 2)

```
TITULO          Decodificador de direcciones
PATRON          Probar 2
REVISION        A
```

FIGURA 9–6 Decodificador de puertos que decodifica los puertos E/S de 8 bits. Este decodificador genera salidas activas en bajos en los puertos F0H hasta F7H.



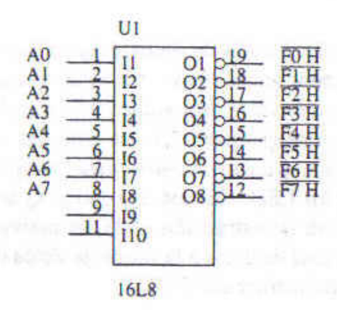


FIGURA 9–7 Decodificador PAL 16L8 que genera señales para los puertos de E/S para F0H–F7H.

EJEMPLO 9–2 (página 2 de 2)

```
AUTOR                 Barry B. Brey
EMPRESA               Symbiotic Systems
FECHA                 6/28/93
CIRCUITO INTEGRADO    Decoder2 PAL16L8

;terminales    1  2  3  4  5  6  7  8  9  10
               A0 A1 A2 A3 A4 A5 A6 A7 NC GND

;terminales    11 12 13 14 15 16 17 18 19 20
               NC F7 F6 F5 F4 F3 F2 F1 F0 VCC

ECUACIONES

/F0 = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * /A0
/F1 = A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * A0
/F2 = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * /A0
/F3 = A7 * A6 * A5 * A4 * A3 * /A2 * A1 * A0
/F4 = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * /A0
/F5 = A7 * A6 * A5 * A4 * A3 * A2 * /A1 * A0
/F6 = A7 * A6 * A5 * A4 * A3 * A2 * A1 * /A0
/F7 = A7 * A6 * A5 * A4 * A3 * A2 * A1 * A0
```

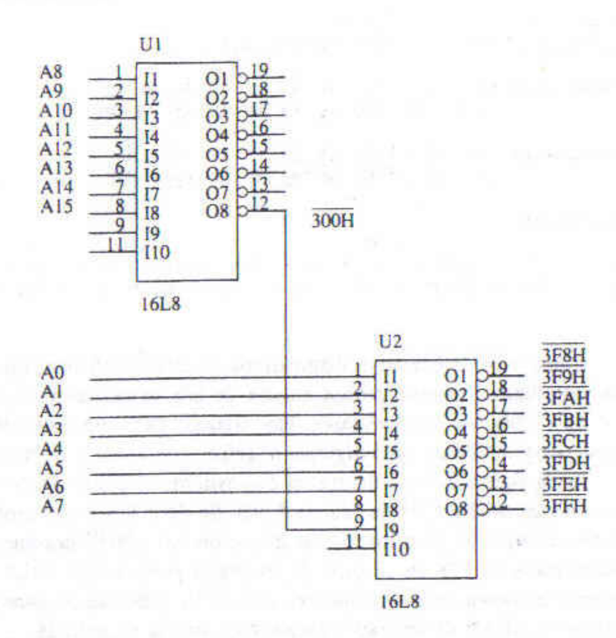
Decodificación de direcciones de E/S de 16 bits

También se decodifican las direcciones de E/S de 16 bits, en especial en un sistema de PC. La diferencia principal entre decodificar una dirección de E/S de 8 bits y una de 16 bits, es que se deben decodificar líneas adicionales (A15–A8) de dirección. En la figura 9-8 se ilustra un circuito que contiene dos PAL 16L8 utilizados para decodificar los puertos de E/S 3F8H–3FFH. Estas son asignaciones comunes para puertos de E/S utilizados en una PC para el puerto de comunicación serial.

EJEMPLO 9–3 (página 1 de 2)

```
TITULO       Decodificador de direcciones
PATRON       Probar 3
REVISION     A
AUTOR        Barry B. Brey
EMPRESA      Symbiotic Systems
```

FIGURA 9–8 Un PAL16L8 que decodifica los puertos de E/S $\overline{3F8H}$ - $\overline{3FFH}$ de 16 bits.



EJEMPLO 9–3 (página 2 de 2)

```
FECHA               6/29/93
CIRCUITO INTEGRADO  Decoder3 PAL16L8

;terminales   1  2  3   4   5   6   7   8   9  10
              A8 A9 A10 A11 A12 A13 A14 A15 NC GND

;terminales   11 12   13 14 15 16 17 18 19 20
              NC 300H NC NC NC NC NC NC NC VCC

ECUACIONES

/300H = /A15 * /A14 * /A13 * /A12 * /A11 * /A10 * A9 * A8
```

El primer PAL 16L8 (U1) decodifica los primeros 8 bits (A15–A8) de la dirección del puerto de E/S, por lo cual genera una señal para habilitar al segundo PAL 16L8 (U2) para cualesquiera direcciones de E/S entre 0300H y 03FFH. El segundo PAL 16L8 también decodifica la dirección de E/S para producir señales de habilitación estroboscópica activas en bajo, $\overline{3F8H}$–$\overline{3FFH}$. Los programas para los PAL U1 y U2 aparecen en los ejemplos 9-3 y 9-4.

EJEMPLO 9–4

```
TITULO              Decodificador de dirección
PATRON              Probar 4
REVISION            A
AUTOR               Barry B. Brey
EMPRESA             Symbiotic Systems
FECHA               6/30/93
CIRCUITO INTEGRADO  Decoder4 PAL16L8
```

```
;terminales  1  2  3  4  5  6  7  8  9    10
             A0 A1 A2 A3 A4 A5 A6 A7 300H GND

;terminales  11 12   13   14   15   16   17   18   19   20
             NC 3FFH 3FEH 3FDH 3FCH 3FBH 3FAH 3F9H 3F8H VCC

ECUACIONES

/3F8H = /300H * A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * /A0
/3F9H = /300H * A7 * A6 * A5 * A4 * A3 * /A2 * /A1 * A0
/3FAH = /300H * A7 * A6 * A5 * A4 * A3 * /A2 * A1 * /A0
/3FBH = /300H * A7 * A6 * A5 * A4 * A3 * /A2 * A1 * A0
/3FCH = /300H * A7 * A6 * A5 * A4 * A3 * A2 * /A1 * /A0
/3FDH = /300H * A7 * A6 * A5 * A4 * A3 * A2 * /A1 */A0
/3FEH = /300H * A7 * A6 * A5 * A4 * A3 * A2 * A1 * /A0
/3FFH = /300H * A7 * A6 * A5 * A4 * A3 * A2 * A1 * A0
```

Puertos de E/S de 8 y de 16 bits

Ahora que se ha entendido que la decodificación de una dirección de puerto de E/S es, quizá, más sencilla que decodificar una dirección en la memoria (debido al número de bits), se explicará cómo se transfieren los datos entre el microprocesador y los puertos de E/S de 8 o de 16 bits. Los datos transferidos a un dispositivo de E/S de 8 bits están en uno de los bancos de E/S en los 8086, 80286 u 80386SX. El sistema de E/S tiene dos bancos igual que la memoria, como se ilustra en la figura 9-9 en la cual se muestran los bancos separados de E/S para un sistema de 16 bits como el de 80286 y 80386SX.

Debido a que hay dos bancos de E/S, cualquier escritura en un dispositivo de E/S de 8 bits, requiere una señal de habilitación estroboscópica separada para escritura para funcionar bien. Las lecturas de dispositivos E/S no requieren señales de habilitación estroboscópica separadas para lectura, porque igual que en la memoria, el microprocesador sólo lee el byte que espera y no tiene en cuenta al otro byte. La única vez en que una señal de lectura puede ocasionar problemas es cuando el dispositivo de E/S responde en forma incorrecta a una operación de lectura. En el caso

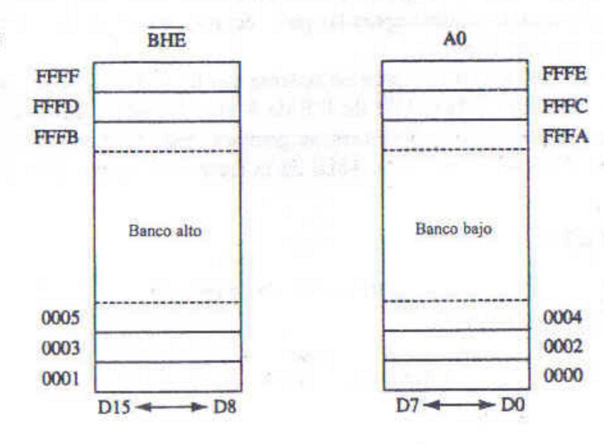


FIGURA 9–9 Bancos de E/S que hay en un sistema basado en el microprocesador 80286.

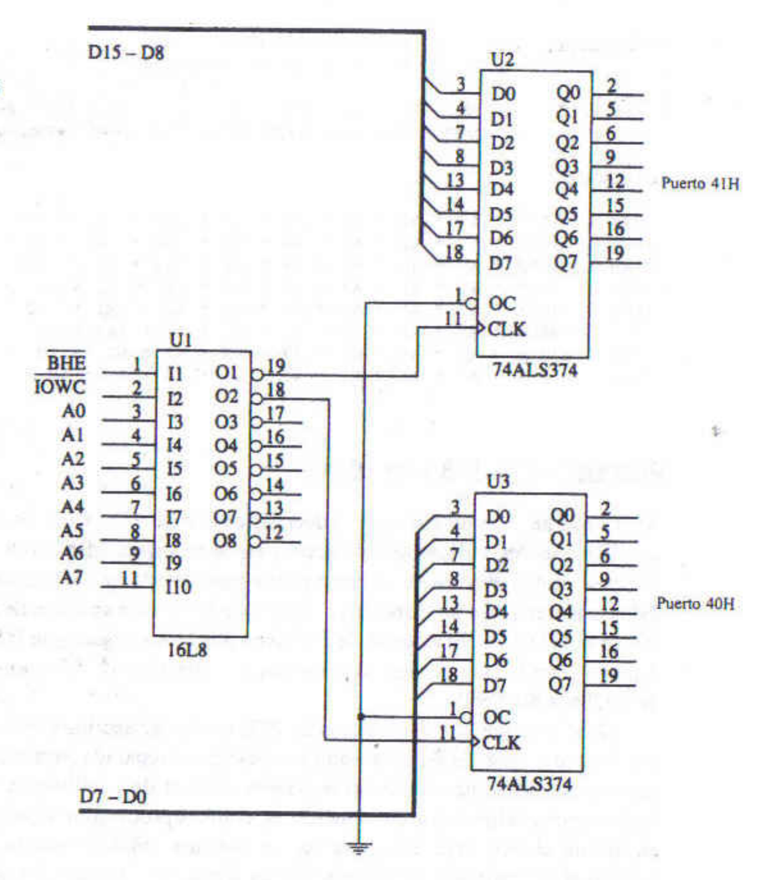


FIGURA 9–10 Decodificador de puertos de E/S que selecciona los puertos 40H y 41H para datos de salida.

de un dispositivo de E/S que responde a una señal de lectura desde el banco incorrecto, puede ser necesario incluir señales separadas para lectura, lo cual se describirá, si se presenta el caso, más adelante en este capítulo.

En la figura 9-10 se ilustra un sistema que tiene dos dispositivos diferentes de salida de 8 bits en una dirección 40H y 41H de E/S de 8 bits. Debido a que estos dispositivos son de 8 bits y porque están en bancos diferentes, se generan señales separadas de escritura para E/S. El programa para el decodificador PAL 16L8 de la figura 9-9 se describe en el ejemplo 9-5.

EJEMPLO 9–5

```
TITULO          Decodificador de dirección
PATRON          Probar 5
REVISION        A
AUTOR           Barry B. Brey
EMPRESA         Symbiotic Systems
```

```
FECHA                    7/1/93
CIRCUITO  INTEGRADO      Decoder5 PAL16L8

;terminales   1   2    3  4  5  6  7  8  9  10
              BHE IOWC A0 A1 A2 A3 A4 A5 A6 GND

;terminales   11 12 13 14 15 16 17 18  19  20
              A7 NC NC NC NC NC NC 040 041 VCC

ECUACIONES

040 = /A0 * /IOWC * /A7 * A6 * /A5 * /A4 * /A3 * /A2 * /A1
041 = /BHE * /IOWC * /A7 * A6 * /A5 * /A4 * /A3 * /A2 * /A1
```

Cuando se seleccionan dispositivos de E/S de 16 bits, las terminales A0 y $\overline{BHE}$ no tienen ninguna función porque ambos bancos de E/S se seleccionan juntos. Aunque los dispositivos de E/S de 16 bits son muy escasos, hay algunos para convertidores analógicos/digitales/y digitales analógicos, así como para algunas interfaces de video y de memoria de disco.

En la figura 9-11 se ilustra un dispositivo de E/S de 16 bits conectado para funcionar en las direcciones 64H y 65H de E/S de 8 bits. Se debe tener en cuenta que el decodificador PAL 16L8 no tiene conexión para los bits de dirección A0 y $\overline{BHE}$ porque estas señales no se aplican en los dispositivos de E/S de 16 bits. El programa para el PAL 16L8 se ilustra en el ejemplo 9-6 para mostrar la forma en que se generan señales de habilitación para registros de acoplamiento de tres estados (4ALS244) utilizados como dispositivos de entrada.

EJEMPLO 9–6

```
TITULO                   Decodificador de dirección
PATRON                   Probar 6
REVISION                 A
AUTOR                    Barry B. Brey
EMPRESA                  Symbiotic Systems
FECHA                    7/2/93
CIRCUITO  INTEGRADO      Decoder6 PAL16L8

;terminales   1    2  3  4  5  6  7  8  9  10
              IORC A1 A2 A3 A4 A5 A6 A7 NC GND

;terminales   11 12 13 14 15 16 17 18 19  20
              NC NC NC NC NC NC NC NC O6X VCC

ECUACIONES

/O6X = /IORC * /A7 * A6 * A5 * /A4 * /A3 * /A2 * /A1
```

Puertos de E/S de 32 bits de ancho

Aunque los puertos de E/S de 32 bits de ancho no son muy comunes, con el tiempo pueden volverse de empleo común debido a los nuevos canales empleados en los sistemas de computadoras. El canal del sistema EISA soporta dispositivos de E/S de 32 bits, pero sólo se encuentra en algunos sistemas de computadora.

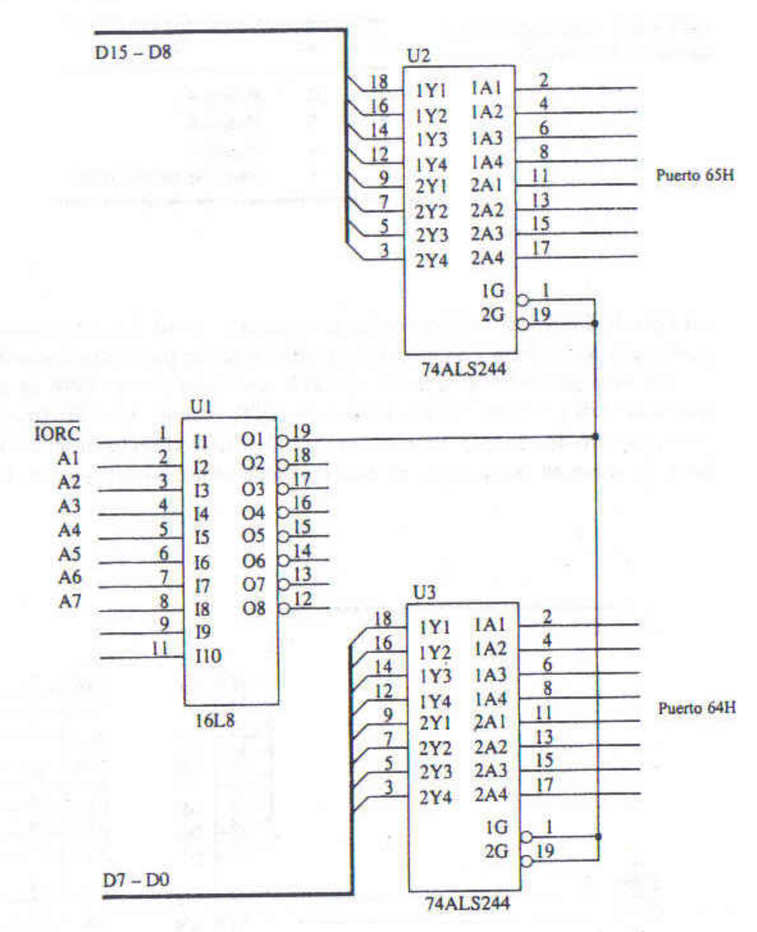


FIGURA 9–11 Un puerto de E/S de 16 bits decodificado en las direcciones 64H y 65H.

El circuito de la figura 9-12 incluye un puerto de entrada de 32 bits para los microprocesadores 80386DX u 80486. Igual que con las interfaces ya citadas, en este circuito se utiliza un solo PAL para decodificar los puertos de E/S y cuatro registros 74LS244 que conectan los datos de E/S con el canal de datos. Los puertos de E/S decodificados con esta interface 70H hasta 73H, de 8 bits, como se explica en el programa para PAL del ejemplo 9-7.

EJEMPLO 9–7

```
TITULO          Decodificador de direcciones
PATRON          Probar 7
REVISION        A
AUTOR           Barry B. Brey
EMPRESA         Symbiotic Systems
FECHA           7/3/93
```

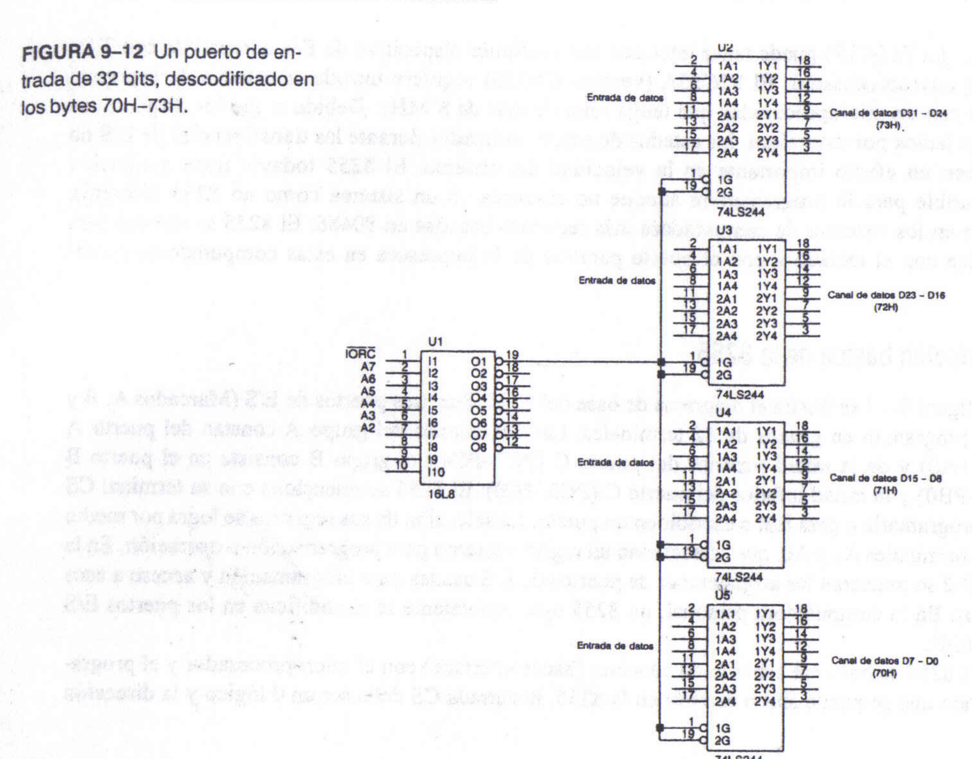


FIGURA 9–12 Un puerto de entrada de 32 bits, descodificado en los bytes 70H–73H.

```
CIRCUITO  INTEGRADO    Decoder7 PAL16L8

;terminales  1    2  3  4  5  6  7  8  9  10
             IORC A7 A6 A5 A4 A3 A2 NC NC GDN

;terminales  11 12 13 14 15 16 17 18 19  20
             NC NC NC NC NC NC NC NC SEL VCC

ECUACIONES

/SEL = /IORC * /A7 * A6 * A5 * A4 * /A3 * /A2
```

9–3 LA INTERFACE PERIFERICA PROGRAMABLE

La interface periférica programable (PPI) 8255 es un muy popular componente de bajo costo para interfaces, que se encuentra en muchas aplicaciones. La PPI tiene 24 terminales para E/S, programables por grupos de 12 terminales, que se utilizan en tres modos diferentes de funciona-

miento. La PPI 8255 puede tener interface con cualquier dispositivo de E/S compatible con TTL para el microprocesador. El 82C55A (versión CMOS) requiere introducir estados de espera si trabaja con un microprocesador que tenga reloj de más de 8 MHz. Debido a que los dispositivos E/S son lentos por naturaleza, los estados de espera utilizados durante las transferencias de E/S no producen un efecto importante en la velocidad del sistema. El 8255 todavía tiene aplicación (compatible para la programación aunque no aparezca en un sistema como un 8255 discreto), incluso en los sistemas de computadora más recientes basados en 80486. El 8255 se emplea para interface con el teclado y con el puerto paralelo de la impresora en estas computadoras personales.

Descripción básica de la 8255

En la figura 9-13 se ilustra el diagrama de base del 8255. Sus tres puertos de E/S (Marcados A, B y C) se programan en grupos de 12 terminales. Las conexiones del grupo A constan del puerto A (PA7–PA0) y de la mitad superior del puerto C (PC7–PC4); el grupo B consiste en el puerto B (PB7–PB0) y la mitad inferior del puerto C (PC3–PC0). El 8255 se selecciona con su terminal CS para programarla o para leer o escribir en un puerto. La selección de sus registros se logra por medio de las terminales A1 y A0, que seleccionan un registro interno para programación u operación. En la tabla 9-2 se muestran las asignaciones de puertos de E/S usadas para programación y acceso a esos puertos. En la computadora personal, un 8255 o su equivalente se decodifican en los puertos E/S 60H–63H.

El 8255 es bastante sencillo de conectar (hacer interface) con el microprocesador y el programa. Para que se pueda leer o escribir en la 8255, la entrada CS debe ser un 0 lógico y la dirección

FIGURA 9–13 Diagrama de base de la interface periférica programable (PPI) 8255A.

TABLA 9-2 Asignación de puertos de E/S para el 8255

A1	*A0*	*Función*
0	0	Puerto A
0	1	Puerto B
1	0	Puerto C
1	1	Registro de comando

correcta de E/S se debe aplicar en las terminales A1 y A0. Las terminales restantes de dirección de puerto son no importa y se decodifican en el exterior para seleccionar 8255.

En la figura 9-14 se muestra un 8255 conectado con el 8086 de modo que funcione como puerto de E/S de 8 bits, en las direcciones C0H (puerto A), C2H (puerto B), C4H (puerto C) y C6H (registro de control). En esta interface se utiliza el sector inferior del mapa de E/S del 8086. Se debe tener en cuenta que en esta interface todas las terminales del 8255 están conectadas

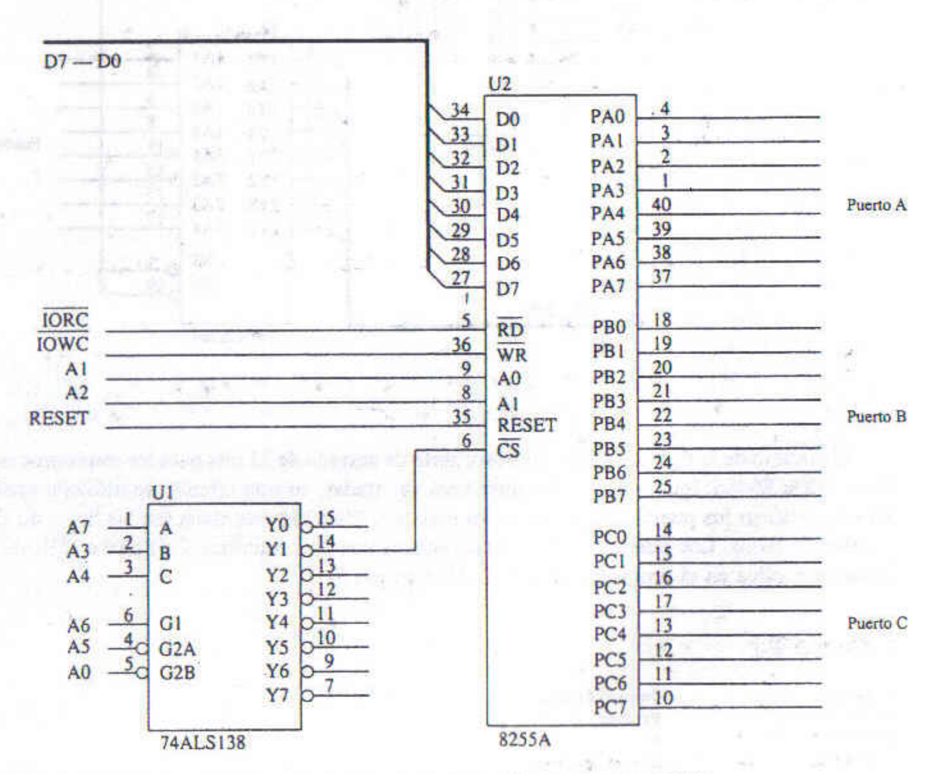


FIGURA 9-14 La 8255A en interface con el banco bajo del microprocesador 80286.

directamente con el 8086, excepto la terminal $\overline{CS}$, la cual se decodifica y selecciona con un decodificador 74ALS138.

La entrada de RESET del 8255 lo inicializa siempre que se inicializa el microprocesador. Una entrada RESET al 8255 hace que se inicialicen todos los puertos como puertos de entrada en el modo de funcionamiento 0. Debido a que las terminales de los puertos son programadas como terminales de entrada, al aplicar un RESET, se evitan daños cuando se aplica corriente por primera vez al sistema. Después de un RESET no se necesitan otras instrucciones para programar el 8255, siempre y cuando se utilice como dispositivo de entrada en los tres puertos. Se debe tener en cuenta que la 8255 tiene interface con la computadora personal en las direcciones 60H–63H de puertos para el control del teclado y también para controlar la bocina, un temporizador y otros dispositivos internos, tales como una expansión de memoria.

Programación del 8255

Es fácil programar el 8255 porque sólo contiene dos posibles comandos básicos, como se ilustra en la figura 9-15. Se verá que el bit de la posición 7 selecciona comandos A o al B. El comando A programa la función del grupo A y B, mientras que el comando B activa (1) bit o desactiva (0) bits del puerto C, sólo si el 8255 se programa en el modo 1 o 2.

Las terminales del grupo B (puerto B y parte inferior del puerto C) se programan como terminales de entrada o de salida. El grupo B puede funcionar en el modo 0 o en el modo 1. El modo 0 es el modo básico de entrada y salida (E/S) que permite programar a las terminales del grupo B como conexiones simples de entrada o de salida con "retención". El modo 1 es el funcionamiento con señales de habilitación estroboscópica en algunos bits del grupo B cuando se transfieren datos por el puerto B y C suministra señales de reconocimiento (handshake).

Las terminales del grupo A (puerto A y parte superior del puerto C) también se programan como terminales de entrada o de salida. La diferencia es que el grupo A puede funcionar en los modos 0, 1 y 2. El funcionamiento en el modo 2 es un funcionamiento bidireccional para el puerto A.

Si se pone un 0 en la posición 7 del byte de comando, se selecciona el comando B. Este comando permite que cualquier bit del puerto C se active (1) o se desactive (0) si se hace funcionar al 8255 en el modo 1 o el modo 2. De lo contrario, este byte de comando no se utiliza. A menudo se utiliza la función de activar o desactivar bits en sistemas de control, para establecer o borrar un bit de control en el puerto C.

Funcionamiento en modo 0

El funcionamiento en modo 0 permite que la 8255 actúe como registro de entrada o como dispositivo de salida con registro transparente. Son lo mismo que los circuitos básicos de entrada y salida descritos en la primera sección de este capítulo.

En la figura 9-16 se ilustra un 8255 conectado con un grupo de ocho dígitos LED de siete segmentos. En este circuito, los puertos A y B están programados como puertos de salida simples con registro transparente (modo 0). El puerto A suministra las entradas de datos de los segmentos a la exhibición visual y el puerto B constituye el medio de seleccionar un dígito a la vez para multiplexar la exhibición visual. El 8255 está en interface con el 8088 por medio de un PAL

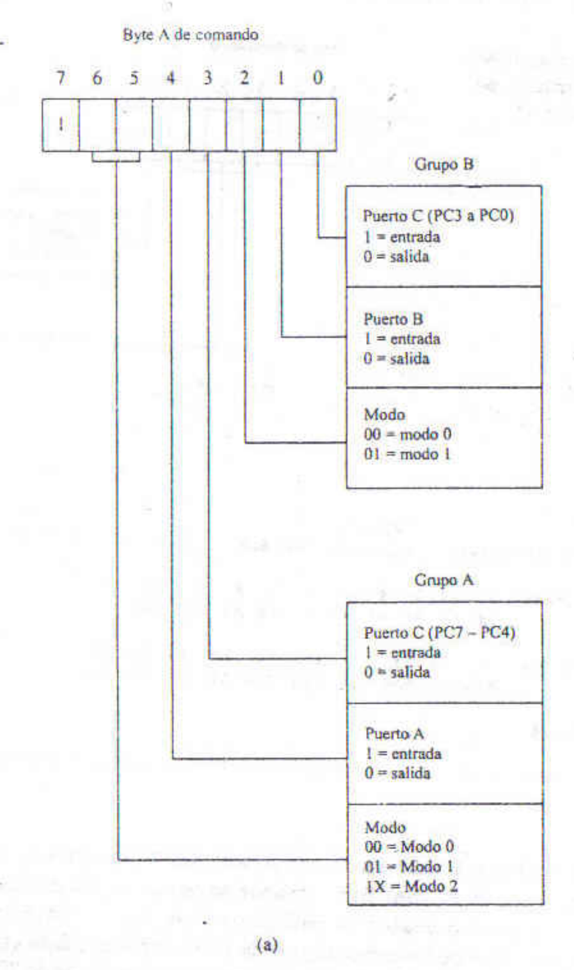


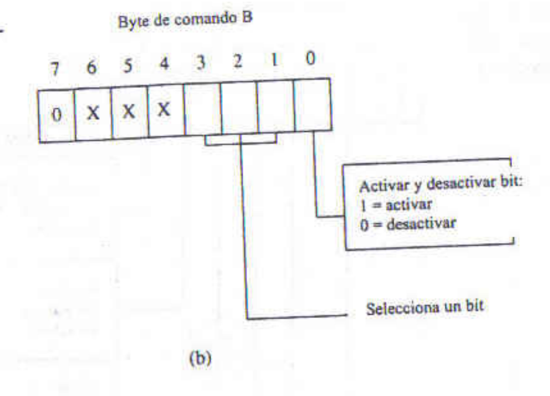
FIGURA 9–15 El byte de comando para el registro de control del 8255A. (a) programación de los puertos A, B y C.

16L8, por lo cual funciona como puertos de E/S con direcciones 0700H–0703H. El programa para el PAL 16L8 se presenta en el ejemplo 9-8. Este PAL decodifica la dirección de E/S y también genera la señal de habilitación estroboscópica de escritura activa en bajo para la terminal $\overline{WR}$ de la 8255.

EJEMPLO 9–8

```
TITULO          Decodificador de dirección
PATRON          Probar 8
REVISION        A
AUTOR           Barry B. Brey
EMPRESA         Symbiotic Systems
```

FIGURA 9–15 (continúa) (b) Activa o desactiva el bit indicado en el campo Selecciona un bit.




```
FECHA                 7/5/93
CIRCUITO  INTEGRADO   Decoder8 PAL16L8

;terminales   1   2   3   4   5   6   7   8   9   10
              A2  A3  A4  A5  A6  A7  A8  A9  A10 GND

;terminales   11  12  13  14  15  16  17  18  19  20
              A11 CS  IOM A12 A13 A14 A15 NC  NC  VCC

ECUACIONES

/CS = /A15*/A14*/A13*/A12*/A11*A10*A9*A8*/A7*/A6*/A5*/A4*/A3*A2*IOM
```

Los valores de los resistores se seleccionan en la figura 9-16 de modo que la corriente para el segmento sea de 80 mA. Esta corriente se requiere para producir una corriente promedio de 10 mA por segmento cuando se multiplexan los dígitos. En este tipo de sistemas de exhibición visual, sólo uno de los ocho dígitos de LED está encendido en un momento dado. La corriente pico del ánodo es de 560 mA, pero la corriente promedio es de 70 mA. Siempre que se multiplexan los dígitos, se aumenta la corriente de segmento de 10 mA hasta un valor igual al número de posiciones de dígitos multiplicado por 10 mA. Esto significa que una exhibición visual de 4 dígitos utiliza 40 mA por segmento, una de 5 dígitos, emplea 50 mA, etc.

Antes de comentar el software para hacer funcionar la exhibición visual, se debe primero programar el 8255, lo cual se logra con la secuencia corta de instrucciones del ejemplo 9-9. Los puertos A y B se programan como salidas.

EJEMPLO 9–9

```
                  ;programa para el 8255A
                  ;
0000 B0 80             MOV   AL,10000000B     ;comando de inicialización
0002 BA 0703           MOV   DX,COMMAND       ;dirección del registro de comando
0005 EE                OUT   DX,AL            ;programar el 8255A
```

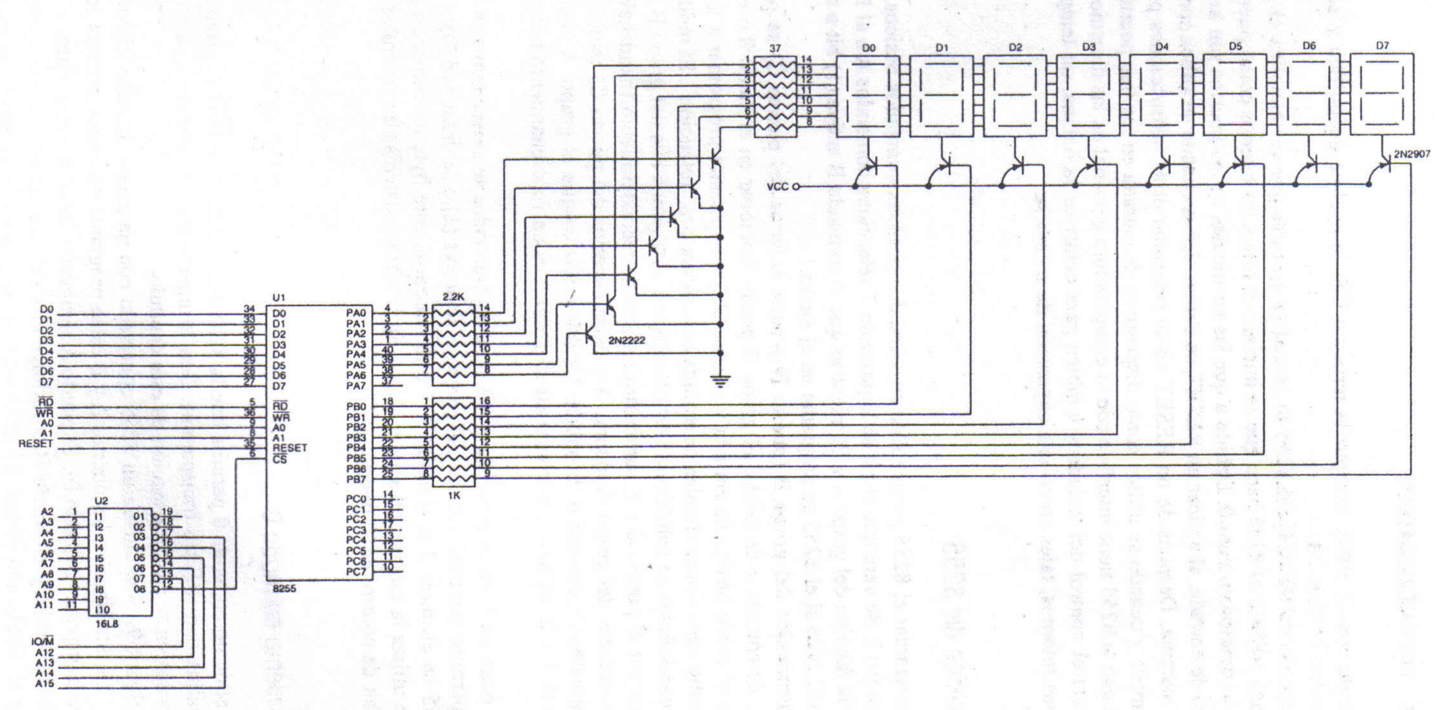


FIGURA 9–16 Una exhibición visual de 8 dígitos LED conectada al microprocesador 8088 por medio de un 8255A.

El procedimiento para excitar esta exhibición se presenta en el ejemplo 9-10. Para el funcionamiento correcto de este sistema de exhibición, hay que llamar a menudo este procedimiento. Se verá que el procedimiento llama al procedimiento Demora que ocasiona un retardo de 1 ms, que no se ilustra en el ejemplo, pero se utiliza a fin de dar tiempo para que se encienda cada posición de la exhibición. Los fabricantes de exhibiciones con LED recomiendan que la multiplexión de la exhibición sea entre 100 Hz y 1500 Hz. Con el empleo de un retardo de 1 ms, se enciende cada LED durante 1 ms para una frecuencia total de exhibición de 1000 Hz/8 o una velocidad de multiplexión de 125.

El procedimiento para la exhibición direcciona una zona de la memoria en la cual están almacenados los datos, en código de 7 segmentos, para los 8 dígitos de exhibición. El registro AH se carga con un código (7FH) que, inicialmente, direcciona la posición más significativa de la exhibición. Una vez seleccionada esta posición, se direccciona al contenido de la localidad MEMORIA + 7 y se envía al dígito más significativo. Después se ajusta el código de selección para la dirección del siguiente dígito de la exhibición. Este proceso se repite 8 veces para exhibir el contenido de las localidades de la memoria de MEMORIA A MEMORIA + 7 en los 8 dígitos de la exhibición visual.

EJEMPLO 9–10

```
                    ;Procedimiento para rastrear los 6 dígitos de la exhibición multiplexada de LED.
                    ;Este procedimiento se debe llamar en forma continua desde un programa para
                    ;exhibir la información codificada de 7 segmentos en la zona MEMORIA.
0006                ;DISP      PROC    NEAR

0006 9C                        PUSHF                          ;salvar los registros
0007 50                        PUSH    AX
0008 53                        PUSH    BX
0009 52                        PUSH    DX
000A 56                        PUSH    SI

                    ;inicializar el registro para la exhibición

000B BB 0008                   MOV     BX,8                   ;cargar contador
000E B4 7F                     MOV     AH,7FH                 ;cargar patrón para selección
0010 BE FFFF R                 MOV     SI,OFFSET MEMORY-1     ;direccionar la RAM de exhibición
0013 BA 0701                   MOV     DX,PORTB               ;direccionar puerto B

                    ;exhibir ocho dígitos

0016                DISP1:

0016 8A C4                     MOV     AL,AH
0018 EE                        OUT     DX,AL                  ;seleccionar dígito
0019 4A                        DEC     DX
001A 8A 00                     MOV     AL,[BX+SI]             ;direccionar puerto A
001C EE                        OUT     DX,AL                  ;obtener datos
                                                              ;exhibir datos
001D E8 029A R                 CALL    DELAY
0020 D0 CC                     ROR     AH,1                   ;esperar 1 ms
0022 42                        INC     DX                     ;ajustar código de selección
0023 4B                        DEC     BX                     ;direccionar puerto B
0024 75 F0                     JNZ     DISP1                  ;ajustar conteo
                                                              ;repetir 8 veces
0026 5E                        POP     SI
0027 5A                        POP     DX                     ;reinicializar registros
```

```
0028 5B                 POP     BX
0029 58                 POP     AX
002A 9D                 POPF
002B C3                 RET

002C          DISP      ENDP
```

Un motor de pasos interface con un 8255. Otro dispositivo que a menudo está en interface con la computadora es un motor de pasos. Este es un motor digital porque se mueve en etapas discretas cuando recorre 360°. Un motor común tiene engranes para moverse, desde quizá, 15° por paso, en un motor de bajo costo, hasta 1° en un motor de pasos más costosos de alta precisión. En todos los casos, estos pasos se generan por medio de varios polos magnéticos o engranes. Se debe tener en cuenta que en esta ilustración hay dos bobinas o devanados excitados. Si se requiere menos corriente, se puede excitar sólo una bobina a la vez, para hacer que el motor funcione en pasos de 0°, 90°, 180° y 270°.

En la figura 9-17 se ilustra un motor de pasos de cuatro bobinas o devanados, el cual tiene una armadura con un solo polo. En esta figura, se ilustra el motor 4 veces, en las que se ha hecho

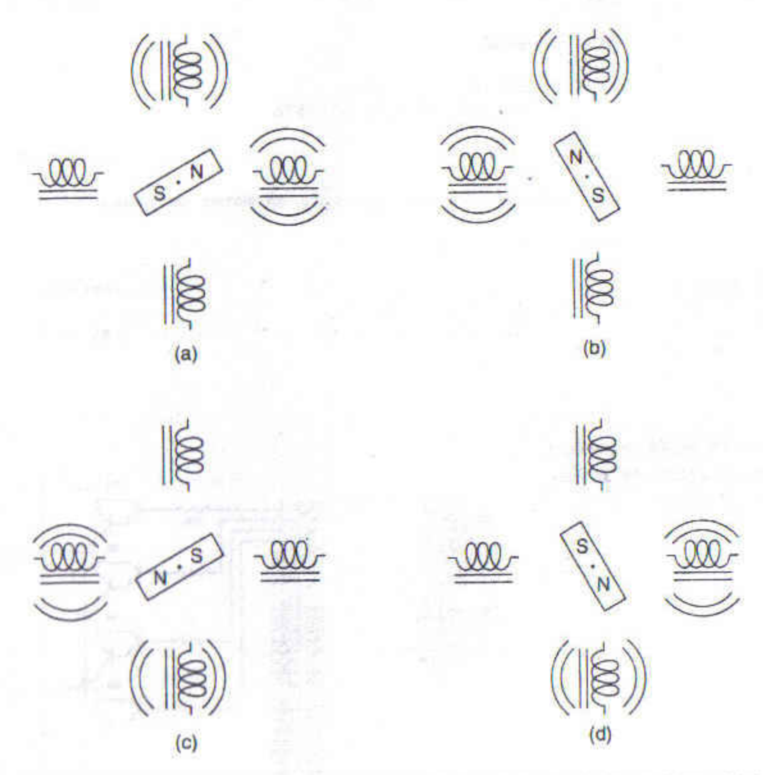


FIGURA 9-17 Motor de pasos mostrando el funcionamiento de un ciclo total: (a) 45°; (b) 135°;(c) 225°; (d) 315°.

girar la armadura (imanes permanentes) a cuatro lugares distintos. Para lograrlo, se excitan los devanados como se ilustra. Se trata de una ilustración de un giro total. El motor de pasos se maneja con pares de transistores Darlington NPN, a fin de proporcionar una corriente elevada en cada devanado.

En la figura 9-18 se ilustra un circuito que puede manejar el motor, y los cuatro devanados se muestran en su lugar. En este circuito se utiliza el 8255 para producir las señales de excitación que hacen girar la armadura del motor hacia la derecha o la izquierda.

Un procedimiento sencillo para excitar el motor (en el supuesto de que el puerto A está programado en el Modo 0 como dispositivo de salida), se presenta en el ejemplo 9-11. A esta subrutina se llama dando, previamente en CX el número de pasos y el sentido de la rotación. Si CX > 8000H, gira a la derecha y si CX < 8000H, gira a la izquierda. Se elimina el bit de CX que está más a la izquierda y los 15 bits restantes contienen el número de pasos. Se debe tener en cuenta que en este procedimiento se utiliza un retardo de tiempo (que no se ilustra) de 1 ms, la cual permite que el motor se mueva a su posición siguiente.

EJEMPLO 9–11

```
0000                SEGMENTO DE DATOS

0000 33             POS        DB      33H                 ;posición

0001                DATA    ENDS

0000                CODE    SEGMENT 'CODE'
                            ASSUME CS:CODE,DS:DATA

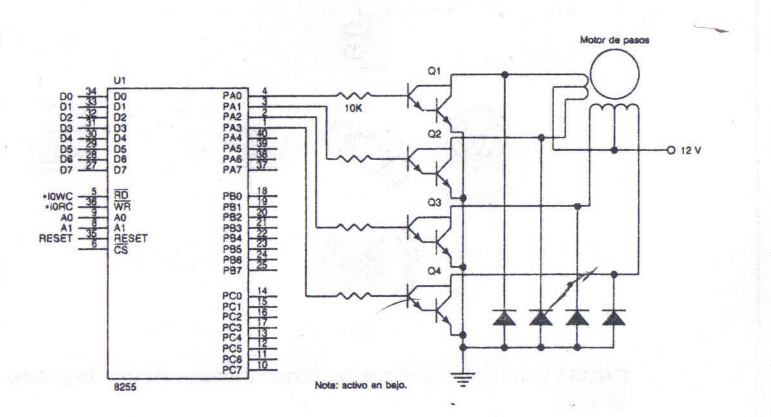
= 0040              PORTA   EQU  40H                       ;número del puerto

                    ;Procedimiento para controlar un motor de pasos

0000                PASO    PROC FAR

0000 A0 0000 R              MOV  AL,POS                    ;obtener posición
0003 81 F9 8000             CMP  CX,8000H
0007 77 10                  JA   RH                        ;si es hacia la derecha (RH)
```

FIGURA 9–18 Motor de pasos e interface con el 8255. No se ilustra el decodificador.



```
0009 83 F9 00          CMP  CX,0
000C 74 14             JE   STEP_OUT              ;si no hay pasos
000E            PASO 1:
000E D0 C0             ROL  AL,1                  ;paso a la izquierda
0010 E6 40             OUT  PORTA,AL
0012 E8 0011           CALL DELAY
0015 E2 F7             LOOP STEP1
0017 EB 09             JMP  STEP_OUT
0019            RH:
0019 81 E1 7FFF        AND  CX,7FFFH              ;borrar bit 15
001D            RH1:
001D D0 C8             ROR  AL,1                  ;paso a la derecha
001F E6 40             OUT  PORTA,AL
0021 E8 0006           CALL DELAY
0024 E2 F7             LOOP RH1
0026            PASO_SAL:
0026 A2 0000           MOV  POS,AL                ;salvar posición
0029 C9                RET

002A            PASO   ENDP
002A            CODIGO ENDS
                       END  STEP
```

La posición actual se almacena en la localidad POS de la memoria, la cual se debe inicializar con 33H, 66H, 99H o CCH. Esto permite una sola instrucción ROR (rotación a la derecha) o ROL (rotación a la izquierda) para hacer girar el patrón de bits binarios al siguiente paso.

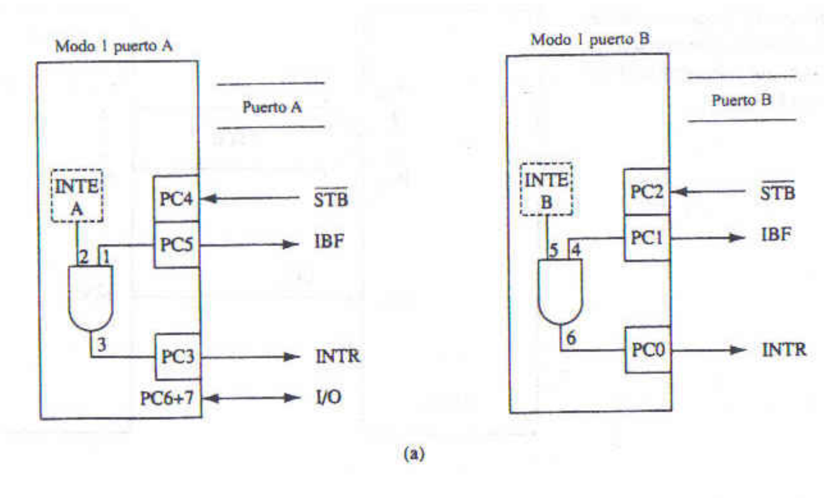
Modo 1: Entrada mediante habilitación

El funcionamiento en el Modo 1 hace que el puerto A o el B funcionen como registros de entrada. Esto permite que los datos externos se almacenen en el puerto hasta que el microprocesador está listo para leerlos. El puerto C se utiliza también en el funcionamiento en Modo 1, no para datos, sino para señales de control o de "reconocimiento" que hacen funcionar al puerto A o al puerto B como puertos de entrada mediante una señal de habilitación estroboscópica. En la figura 9-19 se ilustra la forma en que ambos puertos están estructurados para funcionar mediante una señal de habilitación estroboscópica en Modo 1 y también aparece el diagrama de temporización.

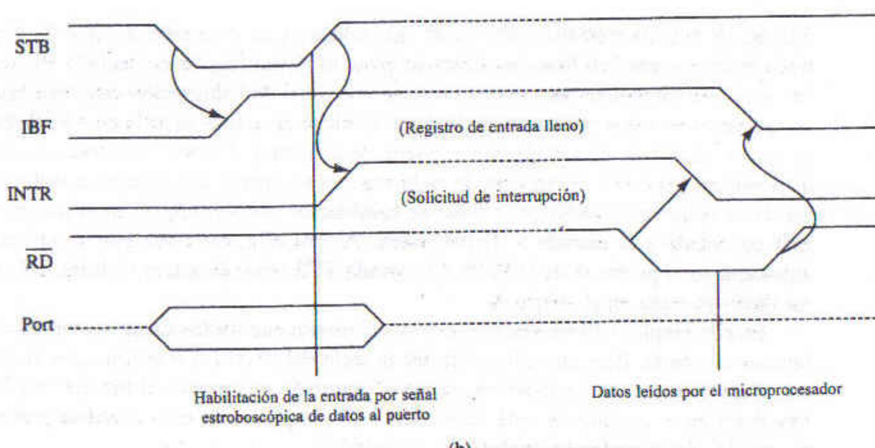
El puerto de entrada mediante la señal de habilitación captura estroboscópica los datos de las terminales del puerto cuando se aplica la señal $\overline{STB}$. Se debe tener en cuenta que esa señal captura los datos del puerto en la transición de 0 a 1. La señal $\overline{STB}$ hace que se capturen los datos en el puerto y también activa las señales IBF (entrada llena) e INTR (solicitar interrupción). Una vez que el microprocesador, por medio de un programa (IBF) o una interrupción (INTR) ha recibido aviso de que hay datos de entrada en el puerto, ejecuta una instrucción IN para leer el puerto ($\overline{RD}$). La acción de lectura del puerto, lleva a IBF y a INTR a su estado inactivo hasta que hay un nuevo dato en el puerto.

Definición de señales para entrada en modo 1

1. $\overline{STB}$. Habilitación estroboscópica: entrada utilizada para cargar datos en el registro del puerto, que retiene la información y se le da entrada al microprocesador por medio de la instrucción IN.



(a)



(b)

FIGURA 9–19 Funcionamiento de entrada habilitada (modo 1) del 8255A. (a) Estructura interna. (b) Diagrama de temporización.

2. IBF. Registro de entrada, lleno. Una salida que indica que el registro de entrada contiene información.
3. INTR. Solicitud de interrupción, es una salida que solicita una interrupción. La terminal INTR se activa con un 1 lógico cuando la entrada $\overline{\text{STB}}$ vuelve al 1 lógico y se desactiva cuando el microprocesador da entrada a los datos del puerto.
4. INTE. Habilitación interrupción ni entrada ni salida, sino un bit interno programado por medio de la posición de bit PC4 (puerto A) o PC2 (puerto B).
5. PC7, PC6. Terminales 7 y 6 de puerto: Son terminales de E/S de uso general que están disponibles para lo que se desee.

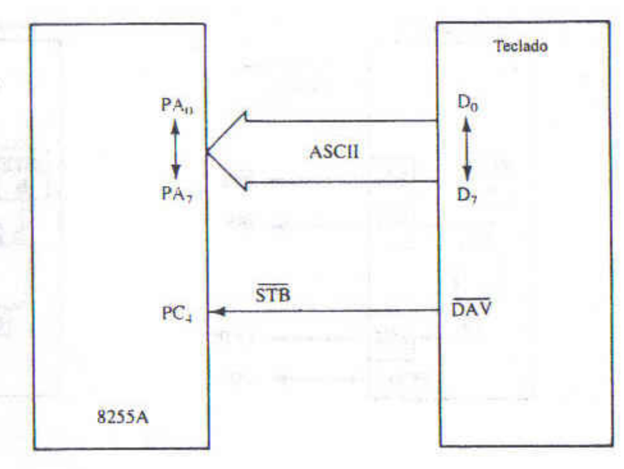


FIGURA 9–20 Empleo del 8255A para funcionamiento por entrada habilitada por señal estroboscópica de un teclado.

Ejemplo de entrada mediante habilitación. El teclado es un excelente ejemplo de dispositivo de entrada mediante una habilitación estroboscópica. El decodificador del teclado elimina los "rebotes" de los interruptores de las teclas y produce una señal de habilitación estroboscópica siempre que se oprime una tecla y que su salida de datos contiene el código de tecla en ASCII. En la figura 9-20 se ilustra un teclado conectado con el puerto de entrada A. En éste, se activa una señal $\overline{DAV}$ (datos disponibles) durante 1 microsegundo cada vez que se oprime una tecla en el teclado. Esto hace que los datos entren por medio de la señal de habilitación estroboscópica en el puerto A, ya que $\overline{DAV}$ está conectada a la entrada $\overline{STB}$ del puerto A. Por ello, cada vez que se oprime una tecla, se almacena en el puerto A de la 8255. La entrada $\overline{STB}$ también activa la señal IBF para indicar que los datos ya están en el puerto A.

En el ejemplo 9-12 se muestra un procedimiento que lee los datos del teclado cada vez que se oprime una tecla. Este procedimiento lee la tecla del puerto A y retorna con el código ASCII en AL. Para detectar una tecla activa, se lee el puerto C y se prueba el bit IBF (bit PC5) para ver si hay datos en el registro. Si está vacío (IBF = 0), el procedimiento continúa probando este bit en espera de que se teclee un carácter en el teclado.

EJEMPLO 9–12

```
                    ;procedimiento que lee el codificador del teclado y retorna
                    ;con el carácter en ASCII a AL
= 0020              BIT5    EQU     20H

0000                LEER    PROC    NEAR

0000 E4 22                  IN      AL,PORTC        ;leer puerto C
0002 A8 20                  TEST    AL,BIT5         ;probar IBF
0004 74 FA                  JZ      READ            ;si IBF = 0

0006 E4 20                  IN      AL,PORTA        ;leer código ASCII
0008 C3                     RET

0009                LEER    ENDP
```

Modo 1: Salida mediante habilitación

En la figura 9-21 se ilustra la configuración interna y el diagrama de temporización del 8255 cuando funciona como un dispositivo con salida por habilitación estroboscópica en Modo 1. El funcionamiento con salida por habilitación estroboscópica es similar a la salida en Modo 0, excepto que se incluyen las señales de control para que haya un protocolo de reconocimiento.

Siempre que se escriben datos en un puerto programado como salida por habilitación estroboscópica, la señal OBF (registro de salida, lleno) se activa en 0 lógico para indicar que los datos están en el registro del puerto. Esta señal indica que los datos están disponibles para un dispositivo de E/S externo que, al tomar los datos activa la entrada $\overline{\text{ACK}}$ (reconocimiento) de habilitación

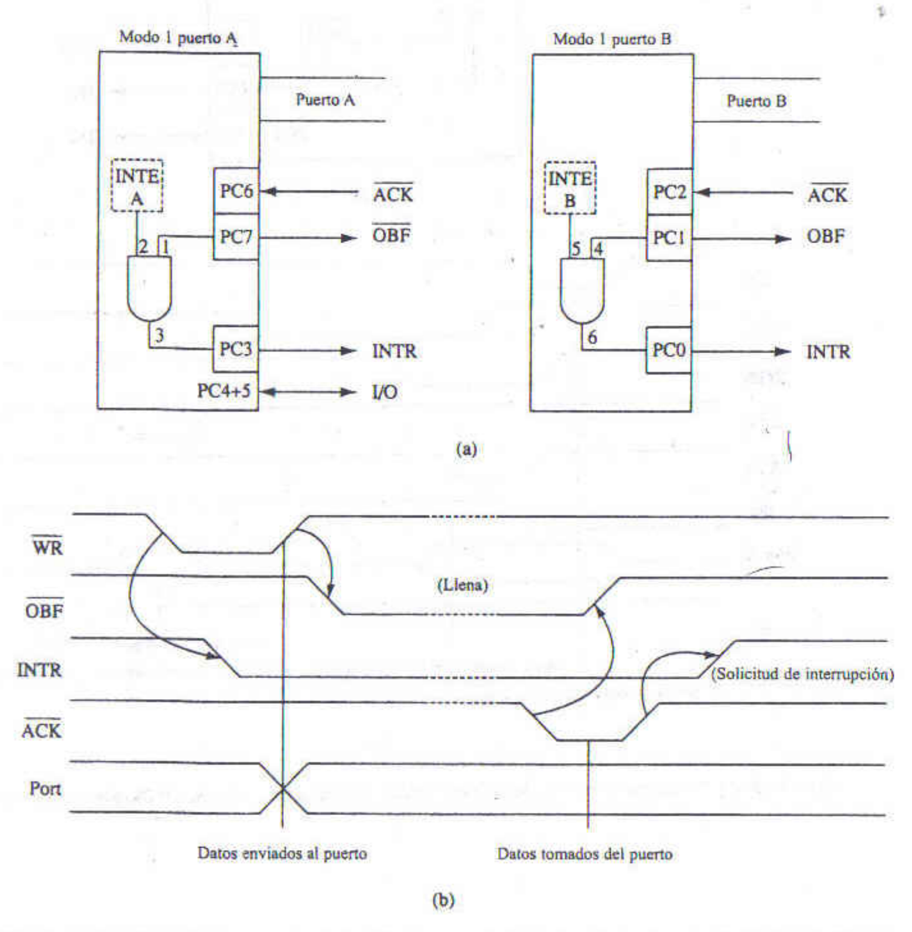


FIGURA 9–21 Funcionamiento por entrada habilitada por señal estroboscópica (modo 1) del 8255A. (1) Estructura interna. (b) Diagrama de temporización.

al puerto. La señal $\overline{ACK}$ desactiva la señal $\overline{OBF}$ otra vez a un 1 lógico para indicar que el registro de salida está vacío.

Definiciones de señales de salida en modo 1

1. $\overline{OBF}$. Registro de salida, lleno: una salida que va abajo siempre que hay salida a datos (OUT) por el puerto A o por el B. A esta señal se desactiva a 1 lógico siempre que el dispositivo externo proporciona el pulso $\overline{ACK}$.
2. $\overline{ACK}$. Reconocimiento: una señal que hace que la terminal $\overline{OBF}$ se desactive con un 1 lógico. La señal $\overline{ACK}$ es la respuesta de un dispositivo externo con la que indica que recibió los datos desde el puerto 8255.
3. INTR. Solicitud de interrupción: señal que interrumpe al microprocesador cuando el dispositivo externo recibe los datos por medio de la señal $\overline{ACK}$. A esta terminal la califica el bit interno INTE (habilitación de interrupción).
4. INTE. Habilitación de interrupción: no es entrada ni salida, sino un bit interno programado para habilitar o deshabilitar a la terminal INTR. El bit INTE A se programa en PC6 y el INTE B en PC2.
5. PC5, PC4. Bits 5 y 4 del puerto C son terminales de E/S para uso general. Las instrucciones para activar y desactivar el bit se puede utilizar para estas dos terminales.

Ejemplo de salida mediante habilitación. En este caso se utiliza la interface con la impresora descrita en la sección 9-1 para demostrar la forma de lograr la sincronización mediante la habilitación de la salida entre la impresora y el 8255. En la figura 9-22 se ilustra el puerto B conectado con una impresora paralela con ocho entradas de datos, para recibir datos en código ASCII: una entrada $\overline{DS}$ (habilitación de datos) para la impresora y una salida $\overline{ACK}$ para reconocer que se recibió de un carácter ASCII.

En este circuito no hay señal para generar la señal $\overline{DS}$ para la impresora, por lo cual se utiliza PC4 con un programa que la genera. La señal $\overline{ACK}$ que regresa la impresora reconoce la recepción de los datos y se conecta con la entrada $\overline{ACK}$ del 8255.

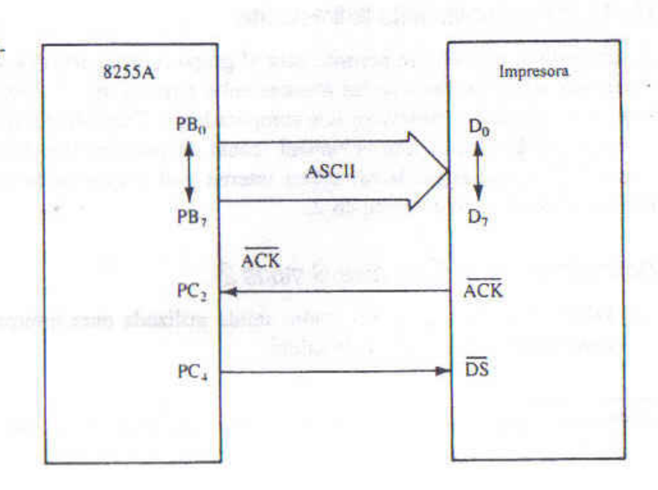


FIGURA 9–22 El 8255A conectado con una interface en paralela que muestra el modo de funcionamiento con salida habilitada por señal estroboscópica del 8255A.

En el ejemplo 9-13 se muestra el programa que envía el caracter en código ASCII en AH a la impresora. El procedimiento prueba primero $\overline{OBF}$ para decidir si la impresora ha recibido los datos del puerto B. Si no, el procedimiento espera la señal $\overline{ACK}$ de la impresora. Si $\overline{OBF}$ = 1, entonces el procedimiento envía el contenido de AH a la impresora por el puerto B y también envía la señal $\overline{DS}$.

EJEMPLO 9–13

```
                        ;Procedimiento que transfiere el carácter en código ASCII de AH
                        ;hasta la impresora por el puerto B
                        ;
= 0002                  BIT1    EQU     2H

0000                    PRINT   PROC    NEAR

                        ;comprobar si la impresora está lista

0000 E4 62                      IN      AL,PORTC        ;obtener OBF
0002 A8 02                      TEST    AL,BIT1         ;probar OBF
0004 74 FA                      JZ      PRINT           ;si OBF = 0

                        ;enviar carácter a la impresora por el puerto B

0006 8A C4                      MOV     AL,AH           ;obtener caracter
0008 E6 61                      OUT     PORTB,AL        ;enviarlo

                        ;enviar DS a la impresora

000A B0 08                      MOV     AL,8            ;borrar DS
000C E6 63                      OUT     COMMAND,AL
000E B0 09                      MOV     AL,9            ;establecer DS
0010 E6 63                      OUT     COMMAND,AL
0012 C3                         RET

0013                    IMPRI   ENDP
```

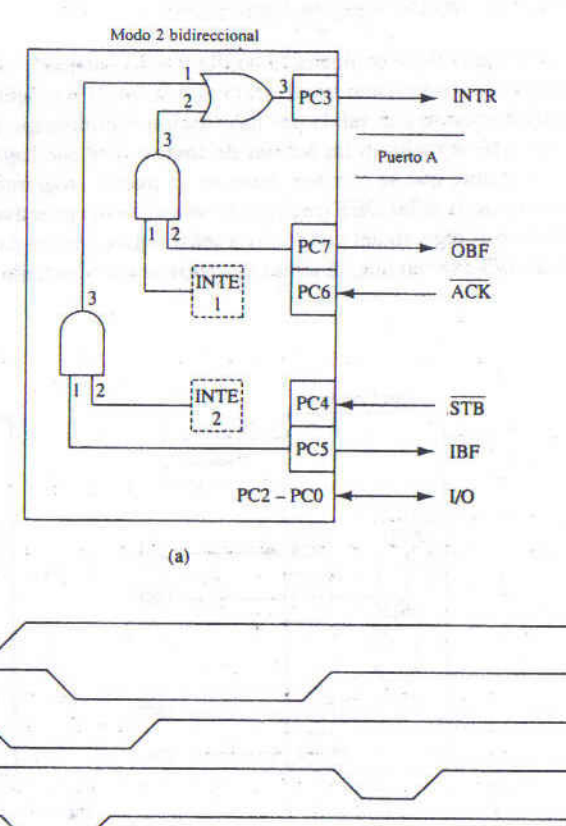
Modo 2: Funcionamiento bidireccional

En el modo 2, que sólo se permite para el grupo A, el puerto A se vuelve bidireccional y permite transmitir y recibir datos en las mismas ocho terminales. Un canal de datos bidireccional es útil cuando se conectan (interface) dos computadoras. También se utilizan para la interface paralela estándar IEEE-488, de alta velocidad (canal de instrumentación de uso general, GPIB[1]). En la figura 9-23 se muestran la estructura interna y el diagrama de temporización para el funcionamiento bidireccional en el modo 2.

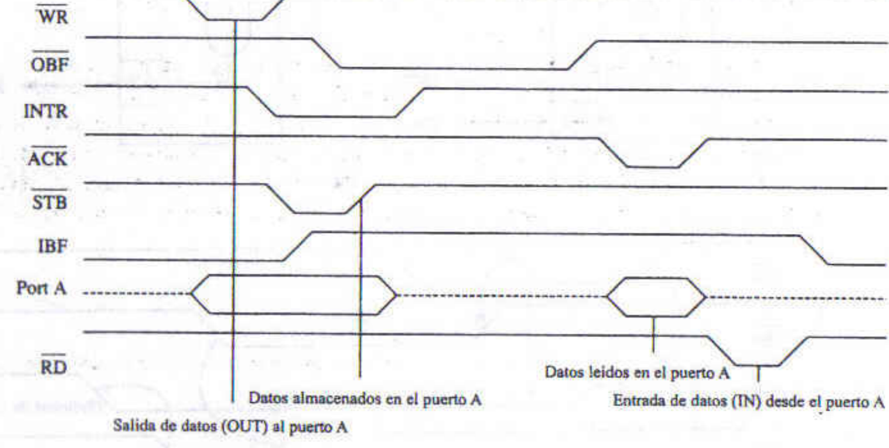
Definiciones de señales para el modo 2

1. INTR. Solicitud de interrupción: salida utilizada para interrumpir al microprocesador para condiciones de entrada y de salida.

[1]GPIB (canal para instrumentos de uso general) es marca registrada de Hewlett-Packard Corporación.



(a)



(b)

FIGURA 9–23 Funcionamiento en Modo 2 del 8255A. (a) Estructura interna. (b) Diagrama de temporización.

2. $\overline{OBF}$: Registro de salida, lleno: una salida que indica que el registro de salida contiene datos para el canal bidireccional.
3. $\overline{ACK}$: Reconocimiento: entrada que habilita los registros de tres estados de modo que los datos puedan aparecer en el puerto A. Si $\overline{ACK}$ es un 1 lógico, los registros de salida del puerto A están en su estado de alta impedancia.
4. $\overline{STB}$. Habilitación estroboscópica: entrada utilizada para cargar el registro de entrada del puerto A con datos externos que vienen del canal bidireccional del puerto A.
5. IBF. Registro de entrada, lleno: salida utilizada para señalar que el registro de entrada contiene datos para el canal bidireccional externo.
6. INTE: Habilitación de interrupción: bits internos (INTE1 e INTE2) que habilitan a la terminal INTR. El estado de la terminal INTR se controla con los bits del puerto C, PC6 (INTE1) y PC4 (INTE2).
7. PC2, PC1 y PC0: Terminales de E/S de uso general en el modo 2, controladas mediante las instrucciones y activación y desactivación de bit.

El canal bidireccional. El canal bidireccional se utiliza con referencia al puerto A con las instrucciones IN y OUT. Para transmitir datos por el canal bidireccional, el programa, prueba primero la señal $\overline{OBF}$ para determinar si el registro de salida está lleno. Si lo está, se envían los datos al registro de salida con la instrucción OUT. Los circuitos externos también monitorean la señal $\overline{OBF}$ para decidir si el microprocesador ha enviado datos al canal. Tan pronto como los circuitos de salida "ven" un 0 en $\overline{OBF}$ envían la señal $\overline{ACK}$ para tomar los datos del registro de salida. La señal $\overline{ACK}$ desactiva al bit $\overline{OBF}$ y habilita a los registros de tres estados de la salida, a fin de poder leer los datos. En el ejemplo 9-14 se presenta un procedimiento que transmite el contenido del registro AH por el puerto A bidireccional.

Para recibir datos por el canal bidireccional del puerto A, el programa prueba el bit IBF para saber si los datos han entrado al registro puerto. Si IBF = 1 entonces se da entrada a los datos con la instrucción IN. La interface externa envía datos hacia el puerto empleando la señal $\overline{STB}$. Cuando se activa $\overline{STB}$ la señal IBF se vuelve un 1 lógico y los datos que hay en el puerto A se "capturan" en un registro de entrada. Cuando se ejecuta la instrucción IN, se desactiva el bit IBF y los datos en el puerto se transfieren a AL. En el ejemplo 9-15 se presenta el procedimiento que lee los datos de ese puerto.

EJEMPLO 9–14

```
                    ;Procedimiento que transmite AH por el canal bidireccional
                    ;del puerto A
                    ;
= 0080              BIT7     EQU     80H

0000                TRANS    PROC    NEAR

                    ;probar OBF

0000 E4 62                   IN      AL,PORTC            ;obtener OBF
0002 A8 80                   TEST    AL,BIT7             ;probar OBF
0004 74 FA                   JZ      TRANS               ;si OBF = 0

                    ;enviar datos

0006 8A C4                   MOV     AL,AH               ;obtener datos
```

```
0008 E6 60              OUT     PORTA,AL
000A C3                 RET

000B            TRANS   ENDP
```

EJEMPLO 9–15

```
                ;Procedimiento que da entrada a datos del canal bidireccional
                ;y torna con ellos en AL
                ;
= 0020          BIT5    EQU     20H

0000            READ    PROC    NEAR

                ;probar IBF

0000 E4 62              IN      AL,PORTC        ;obtener IBF
0002 A8 20              TEST    AL,BIT5         ;probar IBF
0004 74 FA              JZ      READ            ;si IBF = 0

                ;leer datos

0006 E4 60              IN      AL,PORTA
0008 C3                 RET

0009            LEE     ENDP
```

La terminal INTR (solicitud de interrupción) se puede activar desde ambos sentidos del flujo de datos por el canal. Si se habilita INTR con ambos bits INTE, entonces los registros de salida y de entrada producen solicitudes de interrupción. Esto ocurre cuando se hacen entrar los datos mediante una señal estroboscópica a los registros con el empleo de $\overline{STB}$ o cuando se escriben los datos con OUT.

FIGURA 9–24 Resumen de las conexiones de puertos para el 8255A.

		Modo 0		Modo 1		Modo 2
Puerto A		IN	OUT	IN	OUT	I/O
Puerto B		IN	OUT	IN	OUT	No usado
Puerto C	0	IN	OUT	$INTR_B$	$INTR_B$	I/O
	1			IBF_B	$\overline{OBF}_B$	I/O
	2			$\overline{STB}_B$	$\overline{ACK}_B$	I/O
	3			$INTR_A$	$INTR_A$	INTR
	4			$\overline{STB}_A$	I/O	$\overline{STB}$
	5			IBF_A	I/O	IBF
	6			I/O	$\overline{ACK}_A$	$\overline{ACK}$
	7			I/O	$\overline{OBF}_A$	$\overline{OBF}$

Resumen de los modos del 8255

En la figura 9-24 es ilustra un resumen de los tres modos de funcionamiento del 8255. El modo 0 produce E/S sencilla, el modo 1 produce E/S con habilitación estroboscópica y el modo 2 permite E/S bidireccional. Como se mencionó, los tres modos se seleccionan por medio del registro de control del 8255.

9–4 INTERFACE PROGRAMABLE 8279 PARA TECLADO Y EXHIBICION VISUAL

La 8279 es una interface programable para teclado y exhibición visual, que rastrea y codifica hasta 64 teclas y controla una exhibición visual hasta de 16 dígitos. La interface para el teclado tiene una memoria de primero en entrar, primero en salir (FIFO) que le permite almacenar hasta seis tecleos antes de que el microprocesador deba tomar un carácter. La sección de la exhibición visual controla hasta 16 dígitos con una RAM interna de 16 × 8 que almacena la información codificada para exhibición visual.

Descripción básica de la 8279

Como se verá, la 8279 está destinada para facilitar su interface con cualquier microprocesador. En la figura 9-25 se ilustra su diagrama de base y a continuación se define cada una de sus terminales.

Definición de terminales para la 8279

1. A0. Entrada de dirección: una terminal que selecciona datos o control para lectura y escritura entre el microprocesador y la 8279. Un 0 lógico selecciona datos y un 1 lógico selecciona el registro de control o de estado.
2. $\overline{\text{BD}}$. Apagar: una salida utilizada para borrar la exhibición visual.
3. CLK. Reloj: entrada que genera la temporización interna de la 8279. La frecuencia máxima permisible en la terminal CLK es de 3.125 MHz. Otras temporizaciones requieren estados de espera en los microprocesadores, que operan a más de 5 MHz.
4. CN/ST. Control/habilitación: entrada que normalmente está conectada con la tecla control de un teclado.
5. $\overline{\text{CS}}$. Selección de integrado: entrada utilizada para habilitar la 8279 para programación, lectura del teclado e información de estado y escribir datos de control y para exhibición.
6. DB7-DB0. Canal de datos: terminales bidireccionales que se conectan con el canal de datos del microprocesador.
7. IRQ. Solicitud de interrupción: una salida que se activa en 1 lógico cuando se oprime cualquier tecla. Esta señal indica que los datos del teclado están disponibles para el microprocesador.
8. OUTA3–OUTA0. Salidas: utilizadas para enviar datos más significativos a los dígitos de la exhibición.
9. OUTB3–OUTB0. Salidas: utilizadas para enviar datos menos significativos a los dígitos de la exhibición.

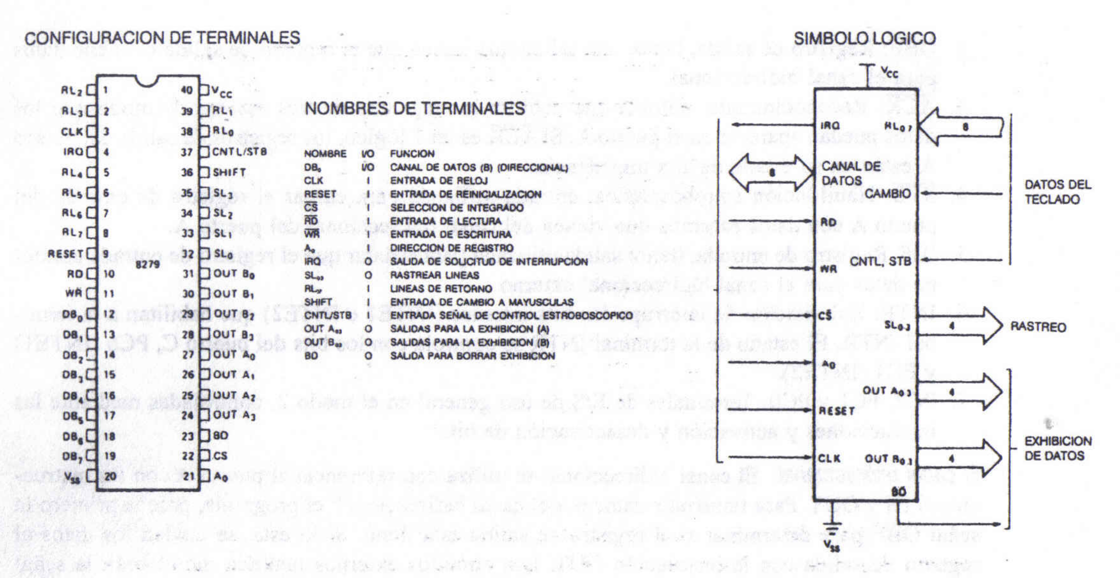


FIGURA 9–25 Diagrama de la base y símbolo lógico de la interface programable 8279 para teclado/exhibición visual. (Cortesía de Intel Corporación.)

10. $\overline{RD}$. Leer: una entrada conectada en forma directa con la señal $\overline{IORC}$ o $\overline{RD}$ del sistema. La entrada $\overline{RD}$ ocasiona, cuando $\overline{CS}$ es un 0 lógico una lectura en los registros de datos o en el registro de estado.
11. RESET. Reinicializa: una entrada que se conecta con la señal RESET del sistema.
12. RL7 hasta RL10. Líneas de retorno: entradas utilizadas para detectar cuándo se oprime cualquier tecla en la matriz de teclado.
13. SHIFT. Cambio a Mayúsculas: una entrada conectada con la tecla o teclas de cambio a mayúsculas en el teclado.
14. SL3 hasta SL0. Líneas de rastreo: salidas que rastrean el teclado y los dígitos de exhibición visual.
15. $\overline{WR}$. Escribir: una entrada que se conecta con la señal de habilitación de escritura que se desarrolla con lógica externa. La entrada $\overline{WR}$ hace que los datos se escriban en los registros de datos o en los registros de control de la 8279.
16. V. Alimentación: terminal conectada a +5.0 volts del canal del sistema.
17. Vss: tierra: terminal conectada con la tierra del sistema.

Interface del 8279 con el microprocesador

En la figura 9-26 la 8279 está conectada con el microprocesador 8088. Se decodifica a la 8279 para que funcione como un dispositivo de E/S de 8 bits en las direcciones 10H y 11H, el puerto 10H es de datos y el 11H es el de control. En este circuito se utiliza un PAL 16L8 (véase ejemplo 9-16) para decodificar la dirección E/S, así como para generar la señal de escritura en el banco inferior de E/S.

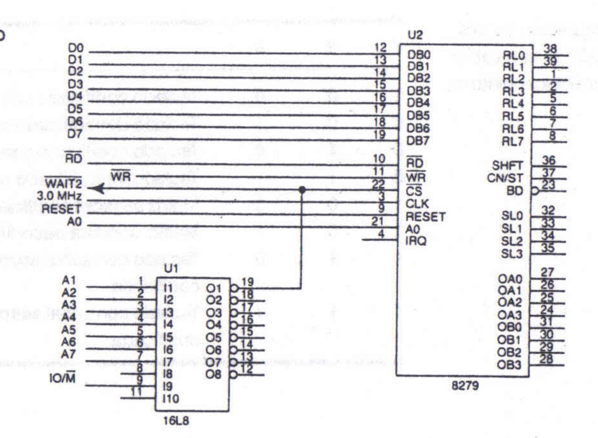


FIGURA 9–26 El 8279 conectado al microprocesador 8088 para funcionar como E/S de 8 bits, puertos 10H y 11H.

El bit A0 del canal de direcciones selecciona el registro de datos o el de control. Se debe tener en cuenta que la señal $\overline{CS}$ selecciona la 8279 y también produce una señal llamada $\overline{WAIT2}$ que se utiliza para generar 2 estados de espera, por lo cual esta interface funciona con un 8088 a 8 MHz.

EJEMPLO 9–16

```
TITULO              Decodificador dirección
PATRON              Probar 9
REVISION            A
AUTOR               Barry B. Brey
EMPRESA             Symbiotic Systems
FECHA               7/5/93
CIRCUITO INTEGRADO  Decoder9 PAL16L8

;terminales  1  2  3  4  5  6  7  8  9   10
             A1 A2 A3 A4 A5 A6 A7 NC IOM GND

;terminales  11 12 13 14 15 16 17 18 19 20
             NC NC NC NC NC NC NC CS NC VCC

ECUACIONES
/CS = IOM * /A7 * /A6 * /A5 * A4 * /A3 * /A2 * /A1
```

La única señal que no está conectada con el microprocesador es la salida IRQ, que es la terminal de solicitud de interrupción y su acción queda fuera del alcance de esta sección. En el capítulo 10 se explican las interrupciones y en dónde funcionan en un sistema.

Interface con el teclado

Supóngase que un teclado de 64 teclas (sin exhibición numérica visual) está conectado con el microprocesador 8088 a través de la 8279. En la figura 9-27 se muestran las conexiones y el

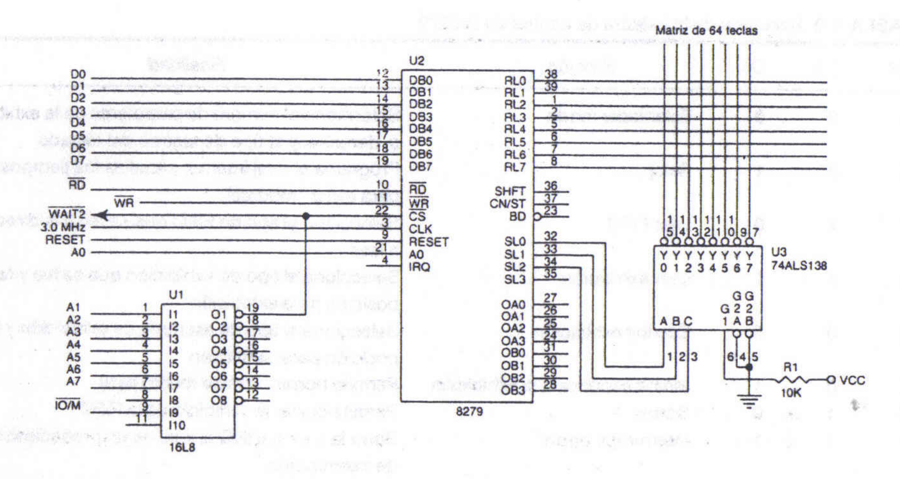


FIGURA 9–27 Un teclado de 64 teclas conectado al microprocesador 8088 por medio del 8279.

teclado. Con la 8279, la matriz del teclado es de cualquier tamaño entre 2 × 2 (4 teclas) y 8 × 8 (64 teclas). (Se debe tener en cuenta que cada punto de cruce de la matriz, contiene un interruptor de botón normalmente abierto, que conecta una columna vertical con un renglón horizontal siempre que se oprime una tecla.)

El número decodificado del puerto de E/S es el mismo que se decodifica para la figura 9-26. El número de puerto de E/S es el 10H para el puerto de datos y 11H para el puerto de control en este circuito.

El decodificador 74ALS138 genera ocho señales, activas en bajo para las columnas del teclado. Las terminales de selección SL2–SL10, rastrean en secuencia cada columna del teclado y los circuitos internos de la 8279 rastrean las terminales RL en busca de una tecla activa. Los resistores que suben el nivel y suelen estar en las líneas de entrada del teclado, no se necesitan, porque la 8279 tiene sus propios resistores internos en las entradas RL.

Programación de la interface del teclado. Antes de poder detectar si se ha oprimido alguna tecla, hay que programar la 8279, lo cual es más complicado que para el 8255. Antes de programar la 8279 hay que tomar en consideración las 8 palabras de control que tiene. Los primeros 3 bits del número enviado al puerto de control (11H en este ejemplo) selecciona a una de las 8 palabras de control diferentes. En la tabla 9-3 aparecen las 8 palabras de control y una breve descripción de cada una.

Descripción de las palabras de control. A continuación aparece una lista de las palabras de control que programan al 8279. Se debe tener en cuenta que los 3 primeros bits son el número del registro de control, que van seguidos por otros bit de información conforme se aplican a cada control.

1. 000DDKKK. Inicializar el Modo: un comando con un código de 000 y dos campos programados para seleccionar el modo de funcionamiento de la 8279. El campo DD selecciona el modo de funcionamiento de la exhibición visual (véase tabla 9-4) y el campo KKK selecciona el modo de funcionamiento (véase tabla 9-5) para el teclado.

TABLA 9–3 Resumen de la palabra de control de la 8279

D7	*D6*	*D5*	*Función*	*Finalidad*
0	0	0	Establecer modo	Selecciona el número de caracteres de la exhibición, izquierda o derecha, y el tipo de rastreo del teclado
0	0	1	Reloj	Programa el reloj interno, inicializa los tiempos de rastreo y para evitar "rebotes"
0	1	0	Leer FIFO	Selecciona el tipo de FIFO que se lee y la dirección que se va a leer
0	1	1	Leer exhibición	Selecciona el tipo de exhibición que se lee y la dirección de la posición para exhibición
1	0	0	Escribir exhibición	Selecciona el tipo de escritura de exhibición y la dirección de posición para exhibición
1	0	1	Inhibir escritura a la exhibición	Permite borrar o inhibir medio byte
1	1	0	Borrar	Permite borrar la exhibición o la FIFO
1	1	1	Interrumpir error!	Borra la terminal IRQ al final de un procedimiento de servicio de interrupción

El campo DD selecciona una exhibición de 8 o de 16 dígitos y determina si se ha dado entrada a datos nuevos en la posición más a la derecha o más a la izquierda de la exhibición. El campo KKK es mucho más complejo. Produce funcionamiento codificado, decodificado o por señales estroboscópicas del teclado.

En el modo codificado, las salidas de SL son activas en altos y siguen la configuración de 0 hasta 7 o de 0 hasta 15, según sea que se seleccionen exhibiciones de 8 o de 16 dígitos. En el modo decodificado, las salidas SL son activas bajas y sólo una de las cuatro salidas es baja en un momento dado. Las salidas decodificadas repiten la configuración o patrón: 1110, 1101, 1011 y 0111. En el modo por señales estroboscópicas un pulso activo en alto en la terminal de entrada CN/ST hace pasar los datos mediante una señal de habilitación estroboscópica desde las terminales RL hasta la FIFO interna en donde se retienen para el microprocesador.

También es posible seleccionar la activación de 2 teclas o bien, de N teclas. La activación para dos teclas impide reconocer dos teclas si se activan al mismo tiempo. La N teclas aceptará todas las teclas oprimidas al mismo tiempo, desde la primera hasta la última.

2. 001PPPPP. Reloj: una palabra de control que programa el divisor interno del reloj. El código PPPPP es un preescalar de pulsos para lograr la frecuencia deseada de funcionamiento o sea,

TABLA 9–4 Asignación de bits binarios para DD de la palabra de control de inicialización de modo

D	*D*	*Función*
0	0	8 caracteres con entrada por la izquierda
0	1	16 caracteres con entrada por la izquierda
1	0	8 caracteres con entrada por la derecha
1	1	16 caracteres con entrada por la derecha

TABLA 9–5 Asignación de bits binarios para KKK de la palabra de control de establecimiento de modo

K	*K*	*K*	*Función*
0	0	0	Teclado codificado con activación de 2 teclas
0	0	1	Teclado decodificado con activación de 2 teclas
0	1	0	Teclado codificado con activación N de teclas
0	1	1	Teclado decodificado con activación de N teclas
1	0	0	Matriz sensora codificada
1	0	1	Matriz sensora decodificada
1	1	0	Teclado con señal estroboscópica; rastreo de exhibición codificada
1	1	1	Teclado con señal estroboscópica; rastreo de exhibición codificada

alrededor de 100 kHz. Por tanto, un reloj de entrada de 1 MHz, requiere un escalador de 01010_2 para PPPPP.

3. 010Z0AAA. Leer la FIFO: palabra de control que selecciona la dirección del código de una tecla en la memoria interna FIFO. Las posiciones AAA de los bits seleccionan la localidad deseada de la FIFO de 000–111 y Z selecciona el modo de autoincremento. En funcionamiento normal, esta palabra de control sólo se utiliza con el funcionamiento de matriz de sensores de la 8279.
4. 011ZAAAA. Leer la exhibición: palabra de control que selecciona la dirección real de una de las localidades de la RAM de la exhibición, para leerla a través del puerto de datos. AAAA es la dirección de la localidad que se va a leer y Z selecciona el modo de autoincremento. Este comando se utiliza si se debe leer información almacenada en la RAM de la exhibición.
5. 100ZAAAA. Escribir la exhibición: una palabra de control que selecciona la dirección de escritura de uno de los caracteres o dígitos de la exhibición. AAAA direcciona la posición en que se va a escribir, a través del puerto de datos y Z selecciona el autoincremento para que las escrituras subsecuentes a través del puerto de datos, sean las posiciones siguientes de la exhibición.
6. 1010WWBB. Inhibir la escritura exhibición: palabra de control que inhibe la escritura en uno u otra mitad de cada localidad de la RAM de la exhibición. La W que está más a la izquierda, inhibe la escritura en los 4 bits que están más a la izquierda de la localidad de la RAM de la exhibición; la W de la derecha inhibe los 4 bits que están más a la derecha. El campo BB funciona en forma similar, salvo que apaga cualquier mitad de las terminales de salida.
7. 1100CCFA. Borrar: palabra de control que borra la exhibición, la FIFO o ambas. El bit F borra la FIFO, el estado de la RAM de la exhibición e inicializa el apuntador de dirección a 000. Si los bits CC son 00 o 01, entonces todas las localidades de la RAM de la exhibición se convierten en 0000000. Si CC = 10, todas las localidades se vuelven 00100000 y si CC = 11, todas las localidades se vuelven 11111111.
8. 111E000. Terminar interrupción: palabra de control que se proporciona para llevar a cero la terminal IRQ en el modo de matriz de sensores.

El gran número de palabras de control, hace que la programación de la interface del teclado parezca compleja. Antes de programar algo, se debe determinar la velocidad del divisor de reloj.

En el circuito ilustrado en la figura 9-26 se muestra una señal de entrada de reloj de 3.0 MHz. Para programar el preescalador de pulsos, con objeto de generar una velocidad interna de 100 kHz, se programa a PPPPP de la palabra de control del reloj, con un 30 o con 111102.

El siguiente paso es la programación del tipo de teclado. En el teclado del ejemplo de la figura 9-27, se tiene un teclado codificado. Se debe tener en cuenta que el circuito incluye un decodificador externo, que convierte los datos codificados que vienen de las terminales SL, en señales decodificadas para selección de columna. En este ejemplo, hay la libertad de escoger ya sea la activación de 2 ON teclas, en la mayor parte de las aplicaciones se utiliza la activación de 2 teclas.

Finalmente, se programa el funcionamiento de la FIFO y una vez que está programada, nunca hay que volver a programarlo, salvo que se necesite leer códigos anteriores del teclado. Cada vez que se oprime una tecla, los datos se almacenan en la FIFO; si se lee el FIFO antes de que éste se encuentre llena (8 caracteres) entonces los datos del FIFO siguen el mismo orden que los datos tecleados. En el ejemplo 9-17 se muestra un programa para inicializar la 8279 y controlar el teclado ilustrado en la figura 9-27.

EJEMPLO 9–17

```
                        ;Programa de inicialización para la interface del teclado
                        ;
                        ;reloj del programa

0000 B0 3E                      MOV  AL,00111110B
0002 E6 11                      OUT  CNTR,AL

                        ;programa de inicialización de Modo

0004 B0 00                      MOV  AL,0
0006 E6 11                      OUT  CNTR,AL

                        ;el programa leerá FIFO y el teclado

0008 B0 50                      MOV  AL,01010000B
000A E6 11                      OUT  CNTR,AL
```

Una vez que se inicializa el 8279, se requiere un procedimiento para leer los datos del teclado. Para determinar un carácter del teclado, se observa el registro de estado de la FIFO. Siempre que se direcciona al puerto de control con la instrucción IN el contenido de la palabra de estado de la FIFO se transfiere al registro AL. En la figura 9-28 se ilustra el contenido del registro de estado de la FIFO y se explica la función de cada uno de sus bits.

EJEMPLO 9–18

```
                        ;Procedimiento que lee datos en FIFO y retorna
                        ;con ellos en AL
                        ;
= 0007                  MASKS   EQU     7

0000                    READ    PROC    FAR

                        ;probar estado de la FIFO
```

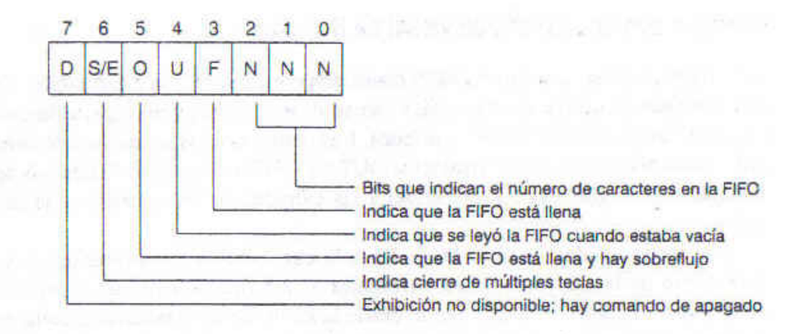


FIGURA 9-28 El registro de estado de FIFO del 8279-5.

```
0000 E4 11              IN   AL,STATUS          ;obtener estado
0002 A8 07              TEST AL,MASKS           ;probar NNN
0004 74 FA              JZ   READ               ;si NNN = 000

                ;leer FIFO

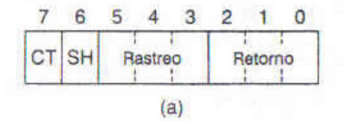
0006 E4 10              IN   AL,DATA            ;obtener datos
0008 CB                 RET

0009            READ    ENDP
```

El procedimiento presentado en el ejemplo 9-18, prueba primero el registro de estado de la FIFO para determinar si contiene algunos datos. Si NNN= 000, la FIFO está vacía. Cuando se determina que la FIFO no está vacía, el procedimiento da entrada de datos a AL y retorna con el código del teclado en AL.

Los datos que se encuentran en AL cuando se retorna del procedimiento, contienen los datos que vienen del teclado. En la figura 9-29 se ilustra el formato de estos datos para los modos de funcionamiento de rastreo y con señales estroboscópicas. El código de rastreo lo entrega la interface del teclado y, para convertirlo a código ASCII, se utiliza la instrucción XLAT junto con una tabla de código ASCII. El código de rastreo entrega el número de renglón y de columna que ocupa los 6 bits más a la derecha. El bit SH muestra el estado de la terminal de cambio a mayúsculas y el bit CT muestra el estado de la terminal de control. En el modo de señales estroboscópicas, el contenido de las ocho entradas RL aparece cuando se les muestrea al colocar un 1 lógico en la terminal de entrada estroboscópica de la 8279.

FIGURA 9-29 El código (a) de rastreo del teclado y (b) código de modo estroboscopio del teclado para la FIFO de la 8279-5.



7	6	5	4	3	2	1	0
RL7	RL6	RL5	RL4	RL3	RL2	RL1	RL0

(b)

Interface con una exhibición visual de 6 dígitos

En la figura 9-30 se ilustran una 8279 conectada con el 8088 y una exhibición visual de 6 dígitos. En esta interface se utiliza un PAL 16L8 (no se ilustra el programa), para decodificar los puertos de E/S 20H (datos) y 21H (control y estado). Los datos para los segmentos se suministran a la exhibición visual por medio de las terminales OUTA y OUTB de la 8279. Estos bits son acoplados con un manejador de segmentos (ULN2003A) a las entradas de los segmentos de los dígitos de la exhibición.

Un decodificador 74ALS138 de 3 a 8 líneas, habilita a los interruptores de los ánodos para cada dígito de la exhibición. Las terminales SL2 hasta SL0 envían al decodificador la posición codificada del dígito de la exhibición desde la 8279. Se debe tener en cuenta que el dígito del lado izquierdo está en la posición 0101 y que el del lado derecho está en la posición 0000. Estas son las direcciones de las posiciones de los dígitos de la exhibición, indicadas en las palabras de control para la 8279.

Es necesario escoger resistores con valores que permitan un flujo de corriente de 60 mA en cada segmento. En este circuito se utilizan resistores de 47 ohms. Si se permite una corriente de segmento de 60 mA, entonces la corriente promedio del segmento es de 10 mA o sea la sexta parte de 60 mA, porque la corriente sólo fluye una sexta parte del tiempo por un segmento. Los interruptores de los ánodos deben suministrar la corriente para los siete segmentos y para el punto decimal. En tal caso, la corriente total de ánodo es de 8 × 60 mA o 480 mA.

EJEMPLO 9–19

```
                      ;Programa de inicialización para la exhibición de 6 dígitos
                      ;
                      ;programar el reloj

0000 B0 3E                      MOV  AL,00111110B
0002 E6 21                      OUT  CNTR,AL

                      ;establecer modo de programa

0004 B0 00                      MOV  AL,0
0006 E6 21                      OUT  CNTR,AL

                      ;borrar exhibición

0008 B0 C1                      MOV  AL,11000001B
000A E6 21                      OUT  CNTR,AL
```

EJEMPLO 9–20

```
                      ;Procedimiento para exhibir AH en la posición
                      ;para exhibición direccionada por AL
                      ;
= 0080                MASKS     EQU     80H

000C                  DISP      PROC    NEAR

000C 50                         PUSH AX              ;salvar datos
000D 0C 80                      OR   AL,MASKS        ;seleccionar exhibición
000F E6 21                      OUT  CNTR,AL
```

FIGURA 9-30 Una exhibición visual numérica de 6 dígitos en interface con un 8279.

```
                    ;exhibir datos

0011 8A C4                  MOV  AL,AH
0013 E6 20                  OUT  DATA,AL
0015 58                     POP  AX
0016 C3                     RET

0017              DISP      ENDP
```

En el ejemplo 9-19 se presenta el programa de inicialización para la 8279 y hace que funcione con esta exhibición de 6 dígitos. Este programa inicializa la exhibición y borra la RAM de la exhibición.

En el ejemplo 9-20 aparece un procedimiento para exhibir información en los dígitos. Los datos se transfieren al procedimiento por medio del registro AX. AH contiene el código de exhibición de 7 segmentos y AL contiene la dirección del dígito a exhibirse.

9–5 TEMPORIZADOR PROGRAMABLE DE INTERVALOS 8254

El temporizador programable de intervalos 8254, consta de tres contadores (temporizadores) programables de 16 bits. Cada contador puede contar en binario o en decimal codificado en binario (BCD). La frecuencia máxima permisible de entrada a cualquier contador es de 10 MHz. Este contador es útil cuando el microprocesador debe controlar eventos en tiempo real. Algunos ejemplos de su empleo son: reloj de tiempo real, contador de eventos y control de la velocidad y sentido de funcionamiento del motor.

Este temporizador también aparece en las computadoras personales en los puertos 40H hasta 43H para (1) generar una interrupción básica del temporizador que ocurre a, más o menos, 18.2 Hz; (2) hacer que se refresque la memoria DRAM del sistema; (3) constituir una fuente de temporización para la bocina interna y otros dispositivos. El temporizador en la computadora personal es un 8253 en vez de un 8254.

Descripción funcional del 8254

En la figura 9-31 se ilustra el diagrama de base del 8254, el cual es una versión de velocidad más alta que el 8253, así como un diagrama de uno de los tres contadores. Cada temporizador tiene una entrada de reloj CLK, una entrada de compuerta, y una conexión de salida (OUT). La entrada de reloj suministra la frecuencia básica de funcionamiento del temporizador, la terminal de compuerta controla al temporizador en algunos modos de operación y en la terminal OUT es donde se obtiene la salida del temporizador.

Las señales que se conectan con el microprocesador son las terminales del canal de datos (D7–D0) $\overline{RD}$, $\overline{WR}$, $\overline{CS}$ y las entradas de direccionamiento A1 y A0. Las entradas de direccionamiento están presentes para seleccionar a cualquiera de los cuatro registros internos utilizados para la programación, la lectura o la escritura en un contador. La computadora personal tiene un temporizador 8253 o su equivalente, decodificado en los puertos de E/S 40H–43H. El temporizador cero se programa para generar una señal de 18.2 Hz que interrumpa al microprocesador en el

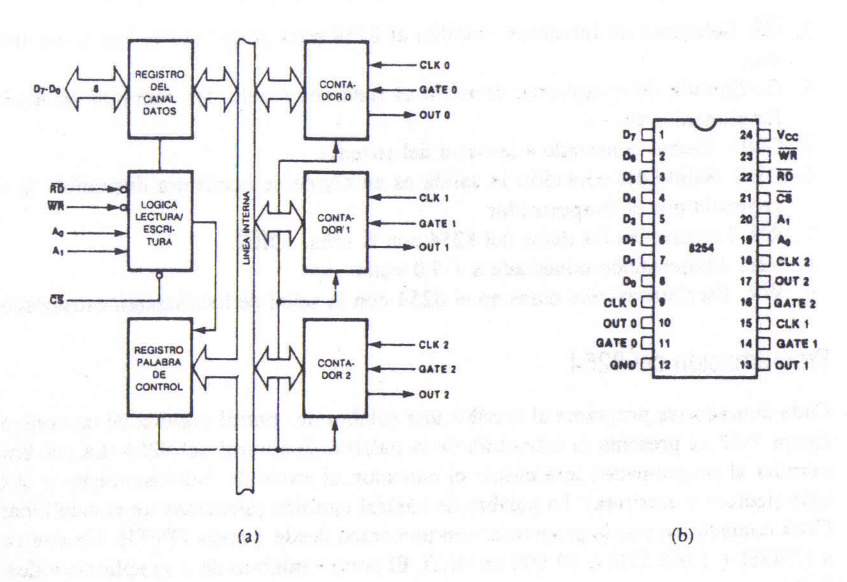


FIGURA 9–31 El temporizador de intervalos programable 8254. (a) Estructura interna. (b) Diagrama de base. (Cortesía de Intel Corporation.)

vector de interrupciones 8 para un "tic" de reloj. El pulso se utiliza a menudo para temporizar programas y eventos. El temporizador 1 se programa para una salida cada 15 microsegundos que se utiliza en la microcomputadora PC/XT para solicitar un ciclo de DMA utilizado para refrescar la RAM dinámica. El temporizador 2 está programado para generar un tono en la bocina de la computadora personal.

Definición de las terminales

1. A1, A0. Entradas de direccionamiento: seleccionan uno de los cuatro registros internos del 8254. En la tabla 9-6 aparecen las funciones de los bits de dirección A1 y A0.
2. CLK. Entrada de Reloj: se utiliza como fuente de temporización para cada uno de los contadores internos. Esta entrada está conectada a menudo con la señal PCLK del controlador de canal del sistema del microprocesador.

TABLA 9–6 Selección de entradas de dirección del 8254

A1	*A0*	*Función*
0	0	Contador 0
0	1	Contador 1
1	0	Contador 3
1	1	Palabra de control

3. $\overline{CS}$. Selección de Integrado: habilita al 8254 para programarlo, leer o escribir en un contador.
4. G. Entrada de compuerta: controla el funcionamiento del contador en algunos modos de funcionamiento.
5. GND. Tierra: conectada a la tierra del sistema.
6. OUT. Salida del contador: la salida es en donde se encuentra disponible la forma de onda generada por el temporizador.
7. $\overline{RD}$. Lectura: lee los datos del 8254 con la señal $\overline{IORC}$.
8. Vcc. Alimentación conectada a + 5.0 volts.
9. $\overline{WR}$. Escribir: escribe datos en el 8254 con la señal de habilitación estroboscópica $\overline{IOWC}$.

Programación del 8254

Cada contador se programa al escribir una palabra de control seguida de un conteo inicial. En la figura 9-32 se presenta la estructura de la palabra de control del 8254. La palabra de control le permite al programador seleccionar el contador, el modo de funcionamiento y el tipo de operación (lectura o escritura). La palabra de control también selecciona un conteo binario o en BCD. Cada contador se puede programar con un conteo desde 1 hasta FFFFH. Un conteo de 0 es igual a FFFFH + 1 (65 536) o 10 000 en BCD. El conteo mínimo de 1 se aplica a todos los modos de funcionamiento, excepto los modos 2 y 3, que tienen un conteo mínimo de 2. El temporizador 0 se utiliza en la computadora personal con un conteo para dividir entre 64K (FFFFH) para generar el pulso de reloj de 18. 2 Hz (18.19638 Hz) para interrupción. El temporizador 0 tiene una frecuencia de entrada de reloj de 4.77 MHz/4 o 1.1925 MHz.

La palabra de control emplea el bit BCD para seleccionar un conteo en BCD (BCD = 1) o un conteo binario (BCD = 0). Los bits M2, M1 y M0 seleccionan uno de los 6 diferentes modos de

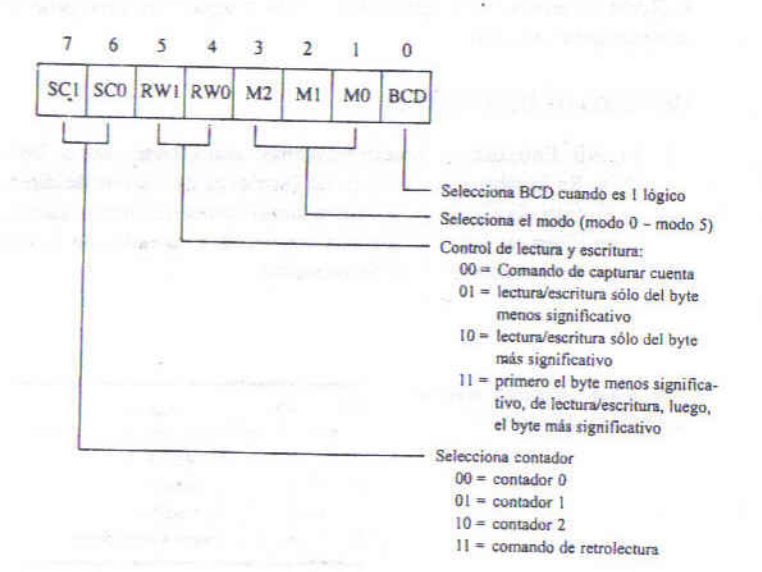


FIGURA 9-32 La palabra de control para el temporizador 8254-2.

funcionamiento (000–101) del contador. Los bits RW1 y RW0 determinan cómo se leerán o escribirán los datos en el contador. Los bits SC1 y SC0 seleccionan un contador o el modo especial de funcionamiento de retrolectura que se describe más adelante en esta sección.

Cada contador tiene una palabra de control que se utiliza para seleccionar la forma en que trabajará el contador. Si se programan 2 bytes en un contador, entonces el primer byte (LSB) detendrá el conteo y el segundo byte (MSB) empezará el conteo con una nueva cuenta. El orden de la programación de cada contador es importante, pero se puede intercalar la programación de diferentes contadores para tener mejor control. Por ejemplo, la palabra de control se puede enviar a cada contador antes que los conteos para la programación de cada uno. En el ejemplo 9-21 se ilustran unas cuantas formas de programar los contadores 1 y 2. En el primer método se programan ambas palabras de control, luego la parte menos significativa del (LSB) del conteo de cada contador, que les detiene su conteo. Después, se programa la parte más significativa (MSB) del conteo, y se ponen a trabajar ambos contadores con el nuevo conteo. En la segunda parte del ejemplo, se muestra un contador programado antes que el otro.

EJEMPLO 9–21

```
PROGRAM CONTROL WORD 1          ;inicializar contador 1
PROGRAM CONTROL WORD 2          ;inicializar contador 2
PROGRAM LSB 1                   ;parar contador 1 y programar LSB
PROGRAM LSB 2                   ;parar contador 2 y programar LSB
PROGRAM MSB 1                   ;programar MSB y arrancar contador 1
PROGRAM MSB 2                   ;programar MSB y arrancar contador 2

    o bien

PROGRAM CONTROL WORD 1          ;inicializar contador 1
PROGRAM LSB 1                   ;parar contador 1 y programar LSB
PROGRAM MSB 1                   ;programar MSB y arrancar contador 1
PROGRAM CONTROL WORD 2          ;inicializar contador 2
PROGRAM LSB 2                   ;parar contador 2 y programar LSB
PROGRAM MSB 2                   ;programar MSB y arrancar contador 2
```

Modos de funcionamiento. Están disponibles seis modos de funcionamiento (modo 0–modo 5) en cada uno de los contadores del 8254. En la figura 9-33 se muestra la forma en que trabajan estos modos con la entrada de reloj (CLK), señal de control de la compuerta (G) o la señal de salida (OUT). A continuación se describe cada modo:

1. El *Modo 0* permite utilizar el contador 8254 como contador de eventos. En este modo, la salida se vuelve un 0 lógico cuando se escribe la palabra de control y permanece allí hasta que hay N más el número de conteo programado. Por ejemplo, si se programa un conteo de 5, la salida seguirá siendo 0 lógico durante seis conteos a partir de N. Se debe tener en cuenta que la entrada de la compuerta (G) debe ser un 1 lógico a fin de permitir el conteo del contador. Si G se vuelve un 0 lógico en la mitad de un conteo, se detendrá el contador hasta que G vuelva a ser otra vez 1 lógico.
2. El *Modo 1* hace que el contador funcione como multivibrador monoestable. En este modo, la entrada G dispara al contador de modo que produzca un pulso en la salida OUT, que se convierte en 0 lógico mientras dura el conteo. Si el conteo es de 10, entonces la salida va bajo durante 10 periodos de reloj cuando se le dispara. Si la entrada G ocurre mientras dura el

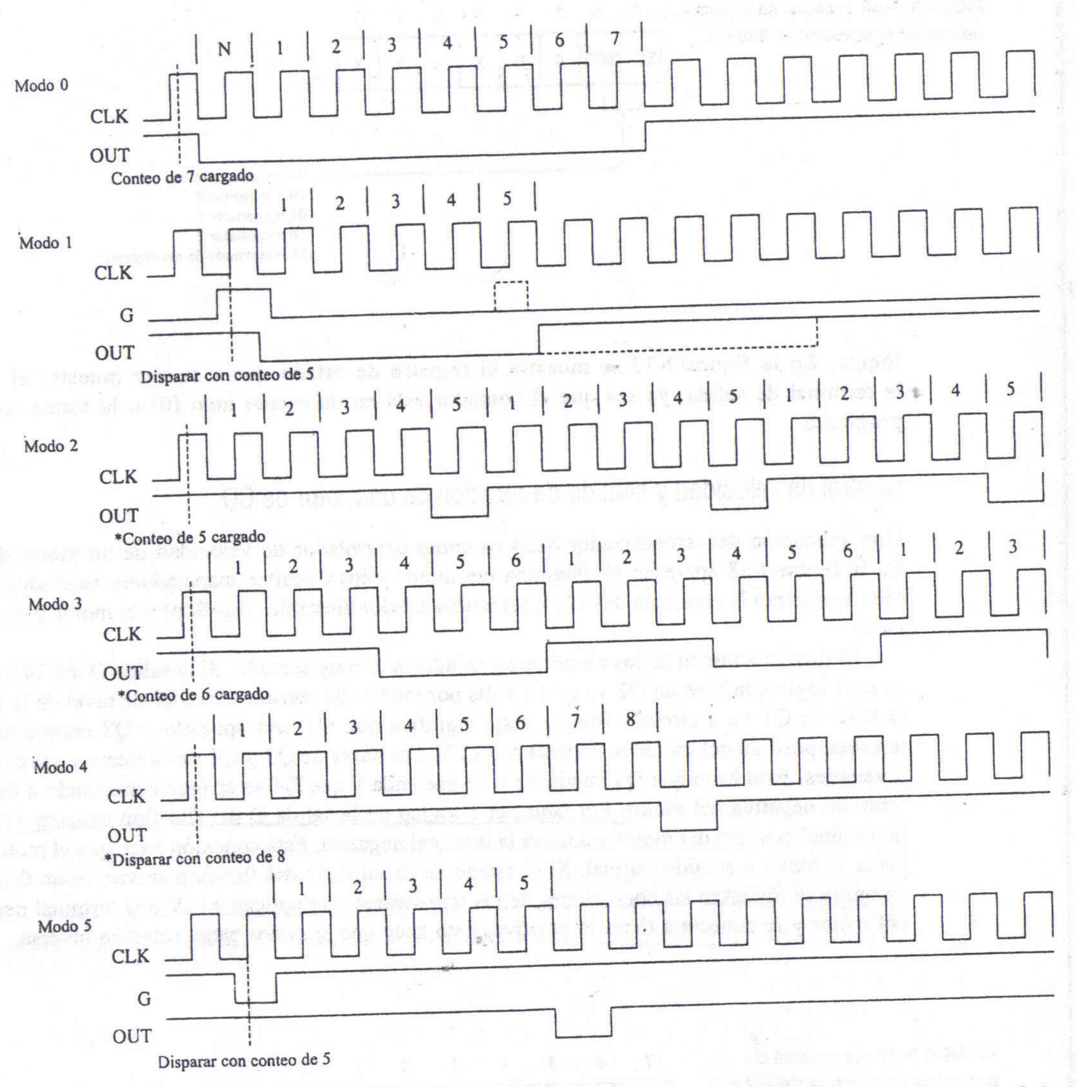


FIGURA 9–33 Los seis modos de funcionamiento del temporizador de intervalos programable 8254-2. *La entrada G detiene el conteo cuando es 0, en los modos 2, 3 y 4.

pulso de salida, se vuelve a cargar el contador con el total de la cuenta y la conexión en la salida continúa mientras dure el conteo.

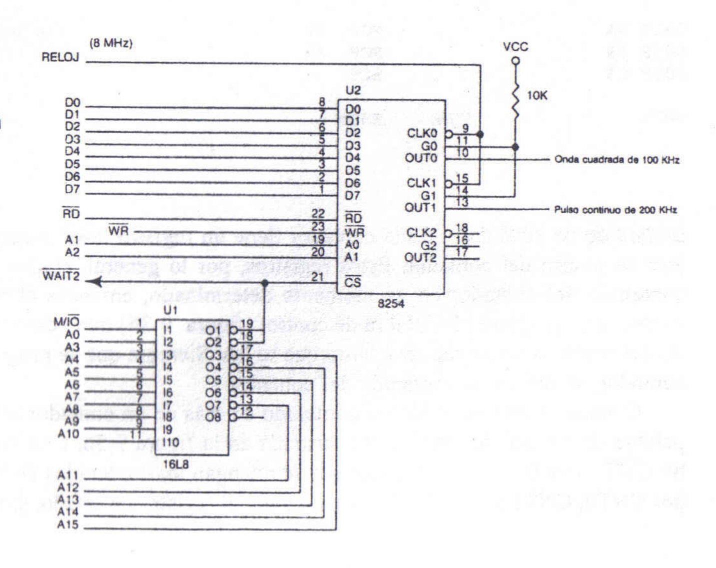
3. El *Modo 2* permite que el contador genere una serie de pulsos continuos que tienen una ancho de un pulso de reloj. La separación entre los pulsos la determina el conteo. Por ejemplo, para

un conteo de 10, la salida es un 1 lógico durante 9 periodos de reloj y está en un nivel bajo durante un periodo de reloj. Este ciclo se repite hasta que se programe el contador con un nuevo conteo o hasta que se lleve la terminal G a un 0 lógico. La entrada G debe ser un 1 lógico para que este modo genere una serie continua de impulsos.

4. El *Modo 3* genera una onda cuadrada continua en la salida, siempre y cuando la terminal G sea un 1 lógico. Si el conteo es par, la salida es alta durante la mitad del conteo y baja durante la otra mitad. Si el conteo es impar, la salida es alta durante un periodo de reloj más largo que aquel en que está baja. Por ejemplo, si se programa el contador para un conteo de 5, la salida está en alto durante 3 pulsos de reloj y en bajo durante 2.
5. El *Modo 4* permite que el contador produzca un solo pulso en la salida. Si se programa un conteo de 10, la salida está en alto durante 10 periodos de reloj y, luego, en bajo durante uno. El ciclo no empieza hasta que el contador tiene su carga completa. Este modo funciona como un multivibrador monoestable disparado por programa. En este modo, igual que en los modos 2 y 3, se emplea la entrada G para habilitar el contador. La entrada G debe ser un 1 lógico para que el contador funcione en estos tres modos.
6. El *Modo 5* es un multivibrador monoestable disparado externamente, que funciona como en modo 4, excepto que lo dispara un pulso de disparo en la terminal G, en vez de que lo haga el programa. Este modo es también similar al modo 1 porque se puede volver a disparar.

Generación de una forma de onda con el 8254. En la figura 9-34, se ilustra un 8254 conectado para funcionar en los puertos de E/S 0700H, 00702H, 0704H y 0706H de un microprocesador 8086. Las direcciones se decodifican con un PAL 16L8 que también genera una señal de habilitación para escritura para el 8254, que está conectado con las conexiones de orden bajo del canal de datos. El PAL también genera una señal de espera para el microprocesador que ocasiona 2 estados de espera cuando se accesa al 8254. El generador de estados de espera conectado con el microprocesador, en la práctica, controla el número de estados de espera insertados en la temporización. El

FIGURA 9–34 El 8254 en interface con un 8086 de 8 MHz, para que genere una onda cuadrada de 100 kHz en OUTO y un impulso continuo de 200 KHz en OUT1.



programa para el PAL no se ilustra aquí, porque es casi el mismo que en muchos de los ejemplos anteriores.

En el ejemplo 9-22 se presenta el programa que genera una onda cuadrada de 100 KHz en OUT0 y un pulso continuo de 200 KHz en OUT1. Se utiliza el modo 3 para el contador 0 y el modo 2 para el contador 1. El conteo programado en el contador 0 es de 80 y el del contador 1 es 40. Estos conteos generan las frecuencias de salida deseadas con un reloj de entrada de 8 MHz.

EJEMPLO 9–22

```
                    ;Procedimiento que programa al temporizador 8254
                    ;para que funcione como se ilustra en la figura 9-34
                    ;
0000                TIME    PROC    NEAR

0000 50                     PUSH AX              ;salvar los registros
0001 52                     PUSH DX

0002 BA 0706                MOV  DX,706H         ;direccionar palabra de control
0005 B0 36                  MOV  AL,00110110B    ;modo 3
0007 EE                     OUT  DX,AL           ;programar control para 0
0008 B0 74                  MOV  AL,01110100B    ;modo 2
000A EE                     OUT  DX,AL           ;programar control para 1

000B BA 0700                MOV  DX,700H         ;direccionar contador 0
000E B0 50                  MOV  AL,80           ;cargar conteo de 80
0010 EE                     OUT  DX,AL
0011 32 C0                  XOR  AL,AL
0013 EE                     OUT  DX,AL

0014 BA 0702                MOV  DX,702H         ;direccionar contador 1
0017 B0 28                  MOV  AL,40           ;cargar conteo de 40
0019 EE                     OUT  DX,AL
001A 32 C0                  XOR  AL,AL
001C EE                     OUT  DX,AL

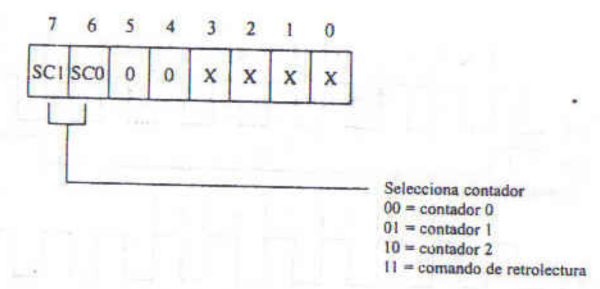
001D 5A                     POP  DX              ;reinicializar registros
001E 58                     POP  AX
001F C3                     RET

0020                TIME    ENDP
```

Lectura de un contador. Cada contador tiene un registro interno que se lee con la operación de leer un puerto del contador. Estos registros, por lo general, siguen el conteo. Si se necesita el contenido del contador en un momento determinado, entonces el registro puede "recordar" el conteo si se programa la palabra de control (figura 9-35) que ocasiona que se retenga el contenido del contador en un registro, hasta que se lea. Siempre que se programa una lectura del registro contador, se retiene el contenido del contador.

Cuando es necesario leer el contenido de más de un contador al mismo tiempo, se utiliza la palabra de control de retrolectura ilustrada en la figura 9-36. Con esa palabra de retrolectura, el bit $\overline{CNT}$ es un 0 lógico para hacer que se retengan los contenidos de los contadores seleccionados por CNT0, CNT1 y CNT2. Si se va a retener el registro de estado, entonces se hace al bit $\overline{ST}$ un 0

FIGURA 9–35 Palabra de control del registro contador del 8254-2.



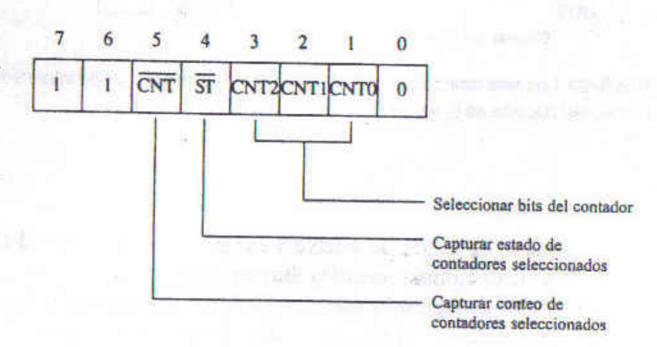
lógico. En la figura 9-37 se muestra el registro de estado que a su vez muestra el estado la terminal de salida, ya sea que el contador está en un estado nulo (0) o la forma como se programó.

Control de velocidad y sentido de rotación de un motor de CD

Una aplicación del temporizador 8254 es como controlador de velocidad de un motor de CD. En la figura 9-38 aparecen el diagrama del motor y los circuitos manejadores asociados. También se ilustran la conexión del 8254, un multivibrador biestable (flip-flop) y el motor y su manejador.

El funcionamiento de los circuitos manejadores es muy sencillo. Si la salida Q del 74ALS112 es un 1 lógico, la base de Q2 va a +12 volts por medio del resistor elevador de nivel de la base y la base de Q1 va a circuito abierto. Esto significa que Q1 está apagado y Q2 encendido y la terminal positiva del motor se conecta a + 12 V. Las bases de Q3 y Q4 van a tierra a través de los inversores. Esto hace que Q3 conduzca o se encienda y que Q4 se apague conectando a tierra la terminal negativa del motor. Por tanto, el 1 lógico en la salida Q del flip-flop conecta +12 V a la terminal positiva del motor y a tierra la terminal negativa. Esta conexión hace que el motor gire hacia el frente o sentido normal. Si el estado de la salida Q del flip-flop se vuelve un 0 lógico, entonces se invierten las condiciones de los transistores y se aplican +12V a la terminal negativa del motor y se conecta a tierra la positiva. Esto hace que el motor tenga rotación inversa.

FIGURA 9–36 La palabra de control de retrolectura del 8254-2.



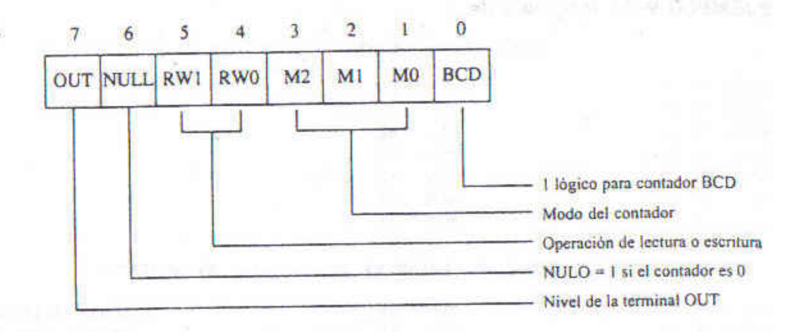


FIGURA 9–37 El registro de estado del 8254-2.

Si se alterna la salida del flip-flop entre un 1 y un 0 lógicos, entonces el motor gira en un sentido u otro a diversas velocidades. Si el ciclo de trabajo de la salida Q es de 50% , el motor no girará y tendrá algo de par (torsión) de retención, porque la corriente pasa por él. En la figura 9-39 se muestran algunos diagramas de temporización y sus efectos en la velocidad y sentido de rotación del motor. Se verá la forma en que cada contador genera pulsos en diferentes posiciones, para variar el ciclo de trabajo en la salida Q del flip-flop. Esta salida se llama también modulación por ancho de pulso.

Para generar estas formas de onda, se programan los contadores 0 y 1, para dividir el reloj de entrada (PCLK) entre 30 720. Para cambiar el ciclo de trabajo de Q se cambia el punto en el cual se pone a funcionar el contador 1 en relación con el contador 0. Con ello, se cambian la velocidad y el sentido de rotación del motor. Pero, ¿por qué dividir el reloj de 8 MHz entre 30 720? Este número es divisible entre 256 por lo cual se puede establecer un programa corto que permita 256 velocidades diferentes. Esto produce también una frecuencia básica de funcionamiento para el motor de alrededor de 260 Hz, que es lo suficiente baja para impulsar el motor. Es importante mantener esta frecuencia de funcionamiento por abajo de 1000 Hz, pero mayor a 60 Hz.

En el ejemplo 9-23 se presenta un procedimiento que controla la velocidad y el sentido de rotación del motor. La velocidad se controla con el valor de AH cuando se llama a este procedimiento. Debido a que se tiene un número de 8 bits para representar la velocidad, entonces un ciclo de trabajo de 50% para un motor parado, es un conteo de 128. Si se cambia el valor en AH cuando se llama a este procedimiento, se puede ajustar o graduar la velocidad del motor. Cuando se llama a este procedimiento y se cambia el valor de AH, se aumentará la velocidad del motor en cualquier sentido de rotación. Conforme el valor de AH se aproxima a 00H, el motor empieza a aumentar su velocidad en la rotación inversa. Cuando el valor de AH se aproxima a FFH, aumenta la velocidad del motor en el sentido normal.

EJEMPLO 9–23 (página 1 de 3)

```
                        ;Procedimiento que controla el ciclo de trabajo de Q y,
                        ;por tanto, la velocidad y sentido de rotación del motor.
                        ;
                        ;AH contiene un número entre 00H y FFH que selecciona
                        ;la velocidad y el sentido de rotación del motor.
                        ;
= 0706                  CNTR    EQU    706H          ;puerto de control
= 0700                  CNT0    EQU    700H          ;puerto 0 del contador
```

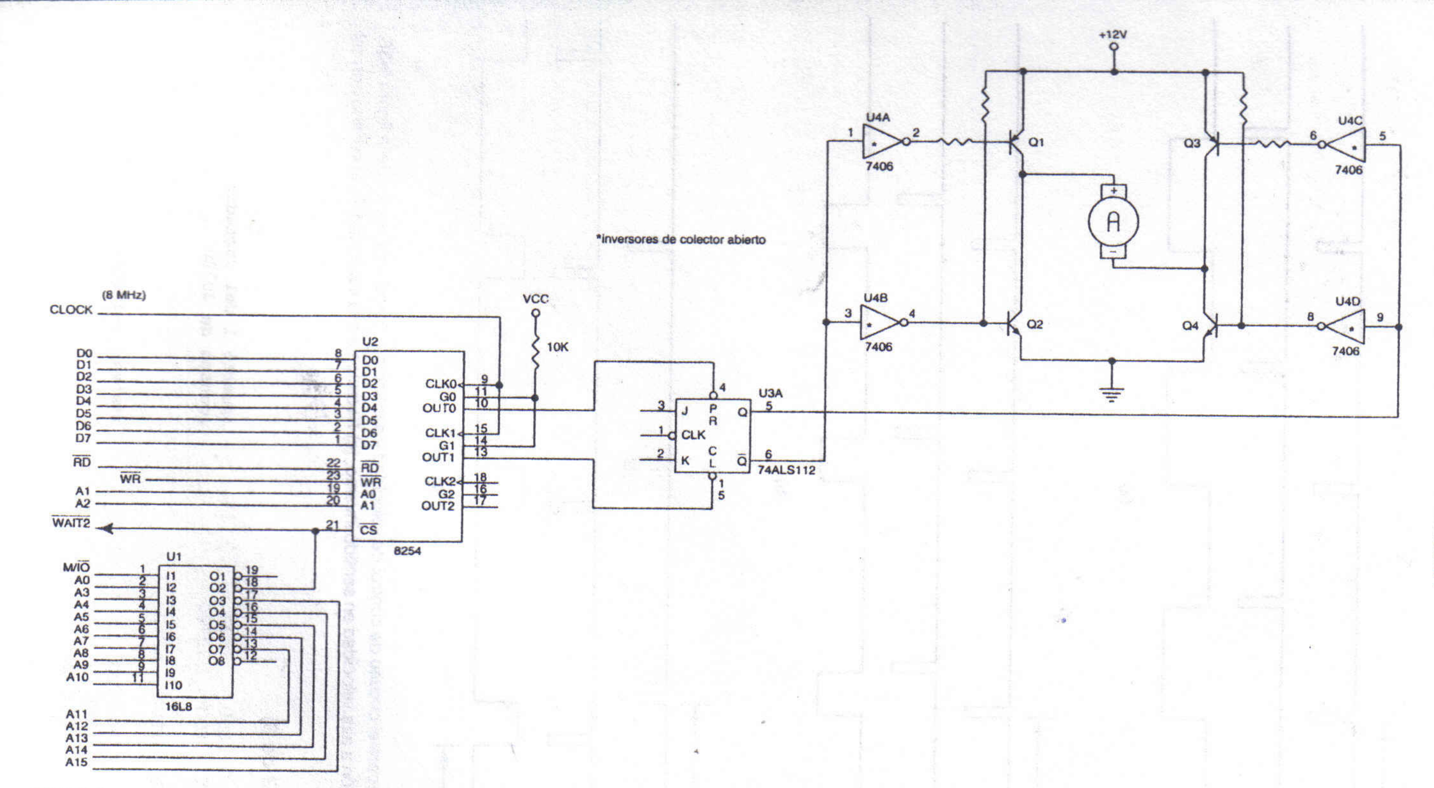


FIGURA 9–38 Control de velocidad y sentido de rotación del motor con el temporizador 8254.

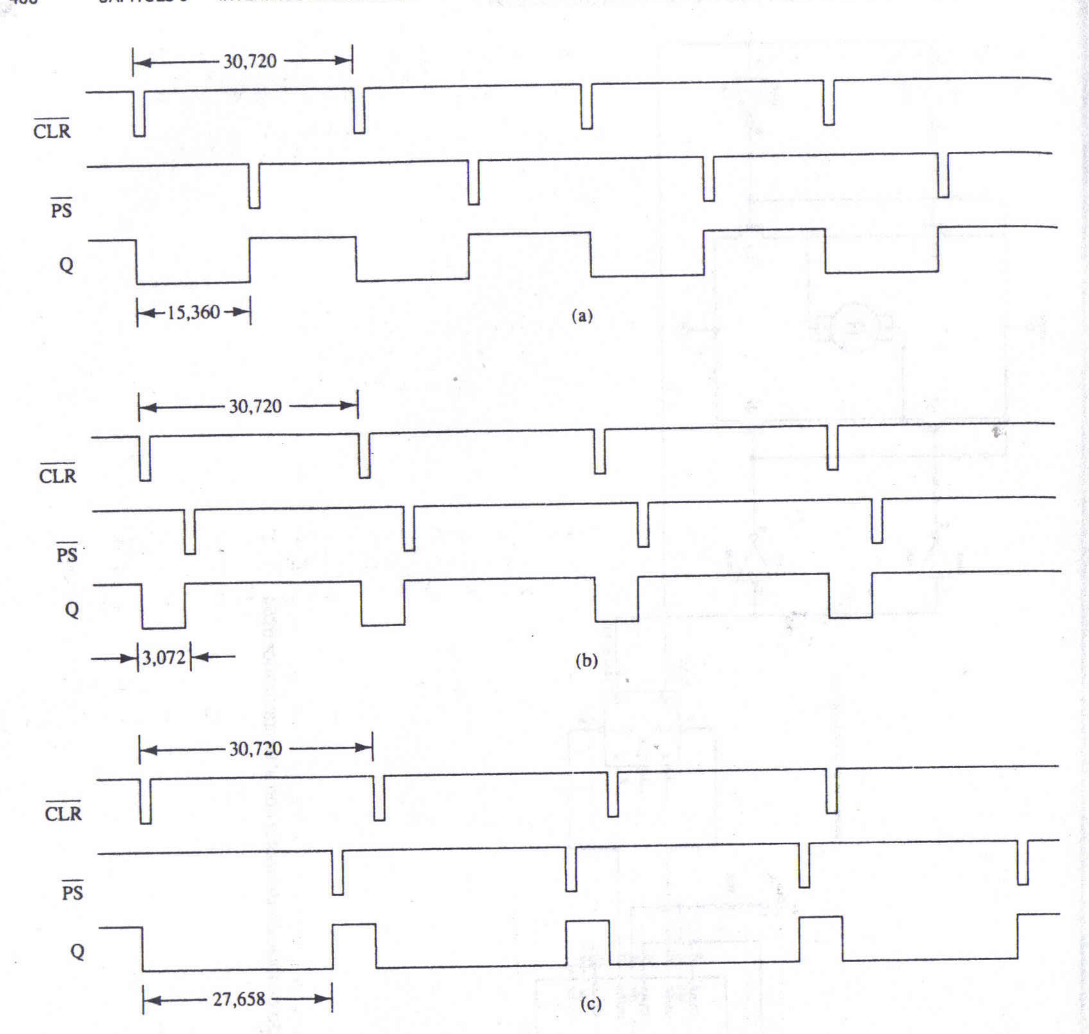


FIGURA 9–39 Temporización para el circuito de control de velocidad y sentido de rotación del motor de la figura 9-38. (a) No hay rotación. (b) Rotación a alta velocidad en sentido inverso. (c) Rotación a alta velocidad en el sentido normal.

EJEMPLO 9–23 (página 2 de 3)

```
= 0702              CNT1    EQU    702H         ;puerto 1 del contador
= 7800              COUNT   EQU    30720        ;conteo de 30720

0000                SPEED   PROC   NEAR

0000 50                     PUSH AX             ;salvar registros
0001 51                     PUSH DX
000253                      PUSH BX
```

EJEMPLO 9–23 (página 3 de 3)

```
                        ;calcular el conteo

0003 8A DC                    MOV  BL,AH
0005 B8 0078                  MOV  AX,120
0008 F6 E3                    MUL  BL
000A 8B D8                    MOV  BX,AX
000C B8 7800                  MOV  AX,COUNT
000F 2B C3                    SUB  AX,BX
0011 8B D8                    MOV  BX,AX

                        ;programa palabras de control del contador

0013 BA 0706                  MOV  DX,CNTR           ;cargar dirección del puerto de control
0016 B0 34                    MOV  AL,00110100B      ;control para CNT0
0018 EE                       OUT  DX,AL
0019 B0 74                    MOV  AL,01110100B      ;control para CNT1
001B EE                       OUT  DX,AL

                        ;arrancar contador 1 para generar borrado

001C BA 0702                  MOV  DX,CNT1           ;direccionar contador 1
001F B8 7800                  MOV  AX,COUNT          ;obtener conteo
0022 EE                       OUT  DX,AL             ;parar contador 1
0023 8A C4                    MOV  AL,AH
0025 EE                       OUT  DX,AL             ;arrancar contador 1

                        ;esperar a que contador 1 llegue al conteo calculado

0026                    SPE:

0026 EC                       IN   AL,DX             ;obtener conteo
0027 86 C4                    XCHG AL,AH
0029 EC                       IN   AL,DX
002A 86 C4                    XCHG AL,AH
002C 3B C3                    CMP  AX,BX             ;probar conteo
002E 72 F6                    JB   SPE               ;si CNT1 está por abajo del conteo

                        ;arrancar contador 0 para generar establecimiento
0030 BA 0700                  MOV  DX,CNT0           ;direccionar contador 0
0033 B8 7800                  MOV  AX,COUNT          ;obtener conteo
0036 EE                       OUT  DX,AL             ;parar contador 0
0037 8A C4                    MOV  AL,AH
0039 EE                       OUT  DX,AL             ;arrancar contador 1

003A 5B                       POP  BX                ;restaurar registros
003B 5A                       POP  DX
003C 58                       POP  AX
003D C3                       RET

003E                    SPEED ENDP
```

Ese procedimiento también ajusta la forma de onda en Q, porque primero calcula la cuenta con la cual debe empezar el contador 0 en relación con el contador 1. Para lograrlo, se multiplica AH por 120 y esa cifra se resta de 30 720. Esto se requiere porque los contadores son de conteo descendente, que cuentan desde el conteo programado hasta 0, antes de volver a trabajar. Luego, se programa el contador 1 con un conteo de 30 720 y se pone a funcionar para generar una onda en cero lógico (borrar) para el flip-flop. Una vez que se pone a funcionar el contador 1, se lee y se

compara con el conteo calculado. Una vez que llega a ese conteo, se pone a funcionar el contador 0 con un conteo de 30 720. Desde ese momento, ambos contadores continúan la generación de las formas de onda de uno y cero, hasta que se vuelve a llamar el procedimiento para ajustar la velocidad y sentido de rotación del motor.

9–6 INTERFACE 8251A PROGRAMABLE PARA COMUNICACIONES

El 8251 es una interface programable para comunicaciones destinada a conectarse con casi cualquier tipo de interface en serial. El 8251A es un transreceptor universal, sincrónico y asincrónico (USART) totalmente compatible con los microprocesadores Intel siempre y cuando se inserten 2 estados de espera cuando funcione a 8 MHz. Para conexión con microprocesadores de velocidad más alta, se deben insertar estados de espera adicionales. El 8251A puede trabajar entre 0 y 64 kBaud (Bd–bits por segundo) en el modo sincrón y entre 0 y 19.2 kBd en el modo asincrón. La frecuencia en baudios es el número de bits transferidos por segundo, que incluyen arranque, paro, datos y paridad. El programador del 8251 selecciona el número de bits de datos, número de bits de paro, tipo de paridad (par o impar) y la velocidad de reloj en el modo sincrón. En el modo asincrón, el programador selecciona el número de bits de datos, paridad y el número de caracteres (uno o dos) para sincronización.

Datos asincrón en serie

Los datos asincrón en forma serial son información que se transmite y recibe sin señal de reloj o temporización. En la figura 9-40 se ilustran dos cuadros de datos asincrón en serie. Cada cuadro contiene un bit de arranque, siete bits de datos, paridad y un bit de paro. En esta figura, un cuadro que contiene un carácter en ASCII tiene 10 bits. En casi todos los sistemas de comunicaciones se utilizan los 10 bits para datos asincrón en serie con paridad par.

Datos sincrón en serie

En la figura 9-41 se ilustra el formato de los datos sincrón en serie. Se verá que esta información no contiene bits de arranque o de paro, sino sólo de datos y paridad. Además, se verá que los datos están sincronizados con un elemento de reloj o temporizador. En lugar de emplear los bits de

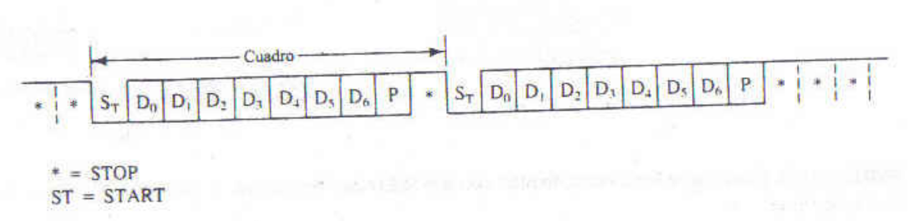
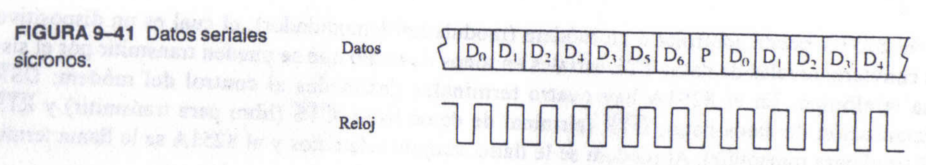


FIGURA 9–40 Datos seriales asíncronos.

FIGURA 9–41 Datos seriales sícronos.



arranque y paro para sincronizar cada caracter, en los datos sincrón se utiliza un caracter o caracteres de sincronización para sincronizar bloques de datos. Ese caracter o caracteres de sincronización se envían seguidos por un bloque de datos en un sistema sincrón. Si se utilizan 2 caracteres de sincronización, el sistema se llama bisincrónico.y es el más común. La ventaja del sistema sincrón es que se transfieren más bits de información por segundo, pero la desventaja es que la señal de reloj también se debe transmitir con los datos.

Descripción Funcional del 8251A

En la figura 9-42 se ilustra el diagrama de base del USART 8251A. Hay dos secciones completamente separadas destinadas a las comunicaciones de datos: el receptor y el transmisor. Debido a que cada sección es independiente de la otra, el 8251A también puede funcionar en los modos símplex, semidúplex y dúplex completo.

Un ejemplo del sistema símplex es cuando el transmisor o el receptor se emplean por sí solos como en una estación de radio de FM (modulación de frecuencia). Un ejemplo del sistema semidúplex es el radio de banda civil, (CB) que transmite y recibe, pero no al mismo tiempo. El sistema dúplex completo permite la transmisión y recepción simultáneas en ambos sentidos; un ejemplo de este sistema es el teléfono.

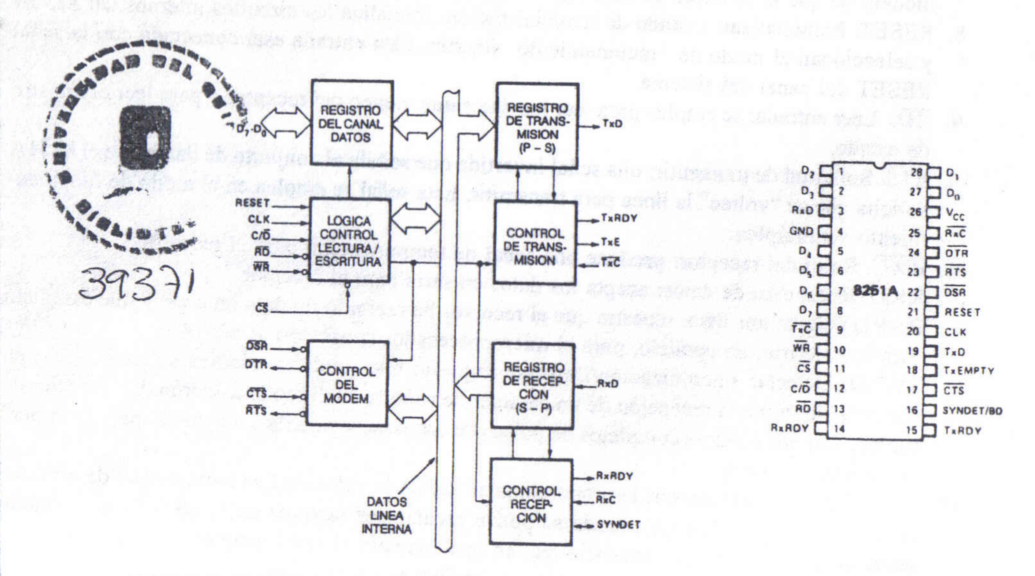


FIGURA 9–42 Diagrama de base y bloques de la interface programable 8251A. (Cortesía de Intel Corporation.)

El 8251A puede controlar a un módem (modulador/demodulador), el cual es un dispositivo que convierte niveles de datos TTL seriales en tonos de audio que se pueden transmitir por el sistema telefónico. En el 8251A hay cuatro terminales destinadas al control del módem: $\overline{\text{DSR}}$ (inicialización de datos lista), $\overline{\text{DTR}}$ (terminal de datos lista), $\overline{\text{CTS}}$ (libre para transmitir) y $\overline{\text{RTS}}$ (solicitud para transmitir). Al módem se le llama conjunto de datos y al 8251A se le llama terminal de datos.

Función de las terminales

1. C/$\overline{\text{D}}$. Comando/Entrada de datos: selecciona el registro de comando/estado o de datos para el transmisor o desde el receptor. Esta terminal es un 1 lógico para accesar al registro de comando/estado y un 0 lógico para accesar a los datos.
2. CLK. Entrada de reloj: le proporciona al 8251A su fuente de temporización. La frecuencia de la señal de reloj debe ser de 3.125 MHz o menos. Esta entrada no determina la velocidad de transmisión de los datos digitales, pero debe ser, cuando menos, 16 veces más alta que los relojes de transmisión o recepción.
3. $\overline{\text{CS}}$. Selección de integrado: habilita al 8251A cuando es 0 lógico.
4. $\overline{\text{CTS}}$. Listo para transmitir: es una señal de retorno del módem que indica que está listo para empezar a enviar información. Esta terminal debe estar conectada a tierra para empezar a transmitir datos.
5. D7 hasta D0. Canal de datos: terminales que se conectan con el canal de datos.
6. $\overline{\text{DSR}}$. Conjunto de datos listo: una entrada inversora utilizada para probar la señal $\overline{\text{DSR}}$ del módem. La entrada $\overline{\text{DSR}}$ indica que el conjunto de datos está listo para empezar la transmisión de información.
7. $\overline{\text{DTR}}$. Terminal de datos lista: entrada inversora que le indica a la terminal de datos del módem de que la terminal de datos (8251A) está lista para transferir información.
8. RESET. Reinicializar: entrada de reinicialización. Inicializa los circuitos internos del 8251A y seleccionan el modo de funcionamiento sincrón. Esta entrada está conectada con la señal RESET del canal del sistema.
9. $\overline{\text{RD}}$. Leer entrada: se emplea para leer los datos que vienen del receptor o para leer el registro de estado.
10. $\overline{\text{RTS}}$. Solicitud de transmitir: una señal invertida que señala al conjunto de datos que el 8251A solicita que se "voltee" la línea para transmitir. Esta señal se emplea en el modo de funcionamiento semidúplex.
11. $\overline{\text{R×C}}$. Reloj del receptor: produce una señal de temporización para el receptor.
12. R×D. Recepción de datos: acepta los datos en serie para el receptor.
13. R×RDY. Receptor listo: muestra que el receptor ha recibido un dato en serie y que está listo para transferirlo, en paralelo, para el microprocesador conectado al 8251A.
14. SY/BD. Detectar sincronización/Detectar Ruptura: una salida que indica sincronización en el modo sincrón o la recepción de un caracter de ruptura en el modo asincrón. Un caracter de ruptura son dos cuadros completos de pulsos de arranque y se utiliza a menudo para "romper" las comunicaciones.
15. $\overline{\text{T×C}}$. Reloj del transmisor: le proporciona al 8251A la velocidad en baudios que determinan la frecuencia del reloj. La entrada se puede escalar por factores de 1, 16 o 64 en el modo asincrón y representa la frecuencia real en baudios para el funcionamiento sincrón.
16. T×D. Transmisión de datos: conexión para salida serial de datos.

17. T×EMPTY. Transmisor vacío: señal que indica que el transmisor del 81251A ha terminado de transmitir todos los datos.
18. T×RDY. Transmisor listo: señal que indica que el transmisor del 8251A está listo para recibir otro caracter para su transmisión.
19. $\overline{WR}$. Entrada de escritura: señal de habilitación estroboscópica para enviar datos al transmisor o al registro interno de comandos para programación.

Programación del 8251A

La programación del 8251 es sencilla al compararla con la de otras interfaces programables descritas en este capítulo. La programación es un trabajo en dos partes que incluye diálogo de inicialización y un diálogo de funcionamiento.

El diálogo de inicialización que ocurre después de una reinicialización por circuito de programa, consta de dos partes: reinicialización y modo. Debido a un aparente error en el diseño, el 8251A no se reinicializa en forma correcta con la terminal de entrada para este propósito (RESET). En vez de ello, se debe reinicializar con una serie de instrucciones que se envían al registro de comando tres 00H seguidos por un 40H (el comando u orden de reinicialización por programa).

Una vez reinicializado se puede programar con la palabra de modo, que lo hace funcionar en forma asíncrona o síncrona. En la figura 9-43 se muestran las palabras comando para los modos síncrono y asíncrono. Ambas palabras especifican el número de bits de datos y de paridad. En el modo asíncrono, se programan el número de bits de paro y el divisor del reloj, y en el modo síncrono se programan el número de caracteres de sincronización y la función de la terminal SY.

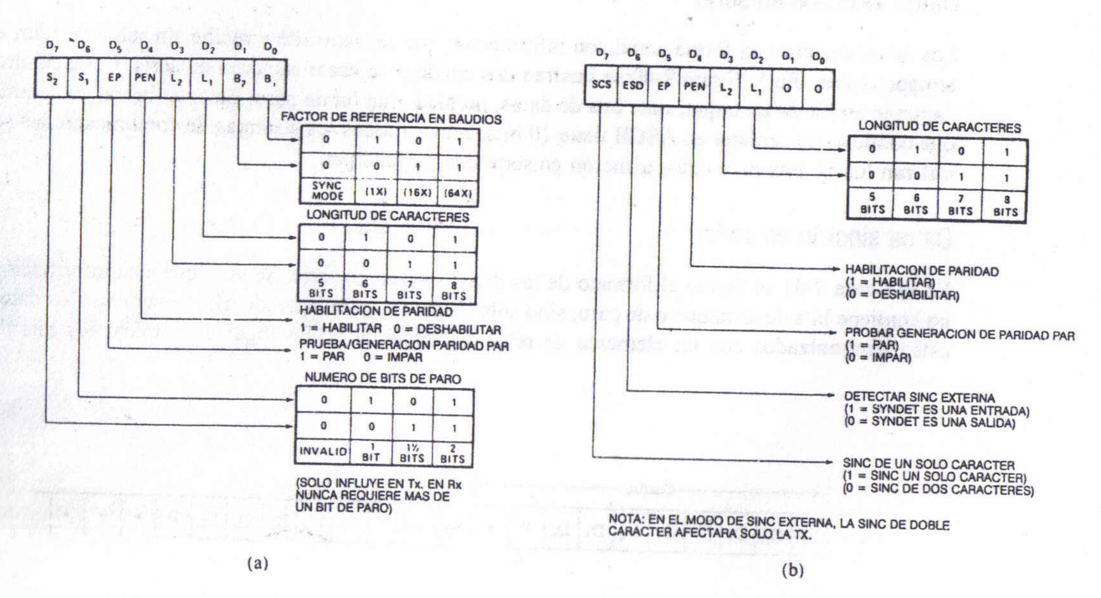


FIGURA 9–43 Palabras de instrucción de modo para el 8251A. (a) Asíncrono. (b) Síncrono. (Cortesía de Intel Corporation.)

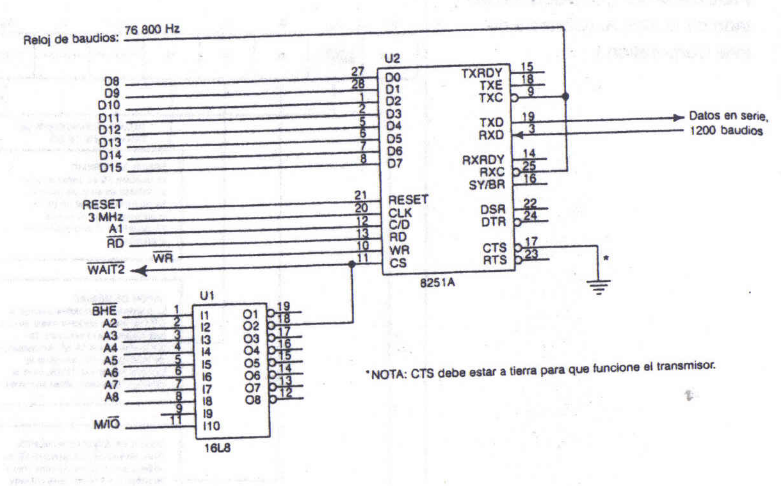


FIGURA 9–44 El 8251A en interface con la parte superior del canal datos del 8086 en los puertos FDH y FFH de E/S.

La terminal SY se programa como entrada, la cual indica sincronización como salida, que indica que hubo recepción de los caracteres de sincronización.

En el modo asíncrono, se programa la instrucción del modo y con ello termina la programación de inicialización. En el modo síncrono, se programa uno o los dos caracteres de sincronización después del comando de modo para concluir la programación de inicialización.

Supóngase que un sistema asíncrono requiere siete bits de datos, paridad impar, un divisor de reloj de 64 y un bit de paro. En el ejemplo 9-24 aparece un procedimiento para inicializar al 8251A para que funcione en esa forma. En la figura 9-44 se muestra la interface con el microprocesador 8086 con el empleo de un PAL 16L8 para decodificar las direcciones de los puertos FDH y FFH de 8 bits. En este caso, el puerto FDH accesa al registro de datos y el FFH al registro de comandos. Se debe tener en cuenta que la velocidad en baudios en la entrada de reloj es de 76 800 Hz y los datos en serie transmitidos y recibidos están a 1 200 baudios. La razón es que el multiplicador de frecuencia en baudios está programado para dividir la entrada de reloj entre un factor de 64 en este ejemplo.

EJEMPLO 9–24

```
                        ;Procedimiento para inicialización del 8251A utilizada en
                        ;funcionamiento síncrono
                        ;
= 00FF                  CNTR    EQU   0FFH              ;puerto para comando
= 0040                  RESET   EQU   40H               ;restablecer código

0000                    PROG    PROC  NEAR

0000 50                         PUSH AX                 ;salvar AX
                        ;restablecer la 8251A

0001 32 C0                      XOR  AL,AL
0003 E6 FF                      OUT  CNTR,AL
0005 E6 FF                      OUT  CNTR,AL
```

```
0007 E6 FF              OUT  CNTR,AL
0009 B0 40              MOV  AL,RESET
000B E6 FF              OUT  CNTR,AL

                  ;programar el modo

000D B0 5B              MOV  AL,01011011B
000F E6 FF              OUT  CNTR,AL

                  ;habilitar el transmisor y el receptor

0011 B0 15              MOV  AL,00010101B
0013 E6 FF              OUT  CNTR,AL

0015 58                 POP  AX                    ;restaurar AX
0016 C3                 RET

0017              PROG  ENDP
```

Si se hace funcionar el 8251A en el modo síncrono, la secuencia de programación es muy semejante, excepto que también se programan uno o los dos caracteres de sincronización. En el ejemplo 9-25 se presenta un procedimiento que programa al 8251A para funcionamiento síncrono, con el empleo de 7 bits de datos, paridad par, SY como salida y dos caracteres de sincronización.

EJEMPLO 9–25

```
                  ;Procedimiento de inicialización para la 8251A utilizada en
                  ;funcionamiento síncrono
                  ;
= 00FF            CNTR    EQU    0FFH              ;puerto para comando
= 0040            RESET   EQU    40H               ;reinicializar código
= 007F            SYNC1   EQU    7FH               ;código de sinc 1
= 007E            SYNC2   EQU    7EH               ;código de sinc 2

0000              PROGS   PROC   NEAR

0000 50                   PUSH AX                  ;salvar AX

                  ;reinicializar el 8251A

0001 32 C0                XOR  AL,AL
0003 E6 FF                OUT  CNTR,AL
0005 E6 FF                OUT  CNTR,AL
0007 E6 FF                OUT  CNTR,AL
0009 B0 40                MOV  AL,RESET
000B E6 FF                OUT  CNTR,AL

                  ;programar el modo

000D B0 B8                MOV  AL,10111000B
000F E6 FF                OUT  CNTR,AL

                  ;características de sincronización del programa

0011 B0 7F                MOV  AL,SYNC1
0013 E6 FF                OUT  CNTR,AL
```

```
0015 B0 7E                  MOV  AL,SYNC2
0017 E6 FF                  OUT  CNTR,AL

                  ;habilitar receptor y transmisor

0019 B0 15                  MOV  AL,00010101B
001B E6 FF                  OUT  CNTR,AL

001D 58                     POP  AX                    ;restaurar AX
001E C3                     RET

001F              PROGS     ENDP
```

Después de programar la instrucción de modo en el 8251A, todavía no está listo para funcionar. Una vez programado el modo de funcionamiento asíncrono y después de programar el modo y los caracteres de sincronización en el modo síncrono, todavía hay que programar el registro de comandos. En la figura 9-45 se ilustra la palabra de comando para el 8251A, la cual habilita al transmisor y al receptor, controla a DTR y RTS, envía un carácter de ruptura en el modo asíncrono, reinicializa errores, reinicializa al 8251A y entra al modo de exploración para funcionamiento síncrono. Consulte el ejemplo de la programación de las palabras de comando en los ejemplos 9-24 y 9-25.

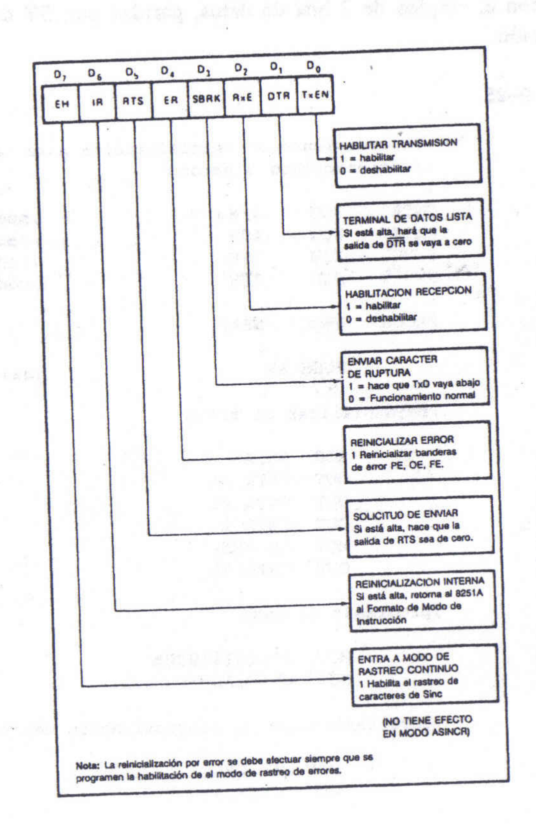
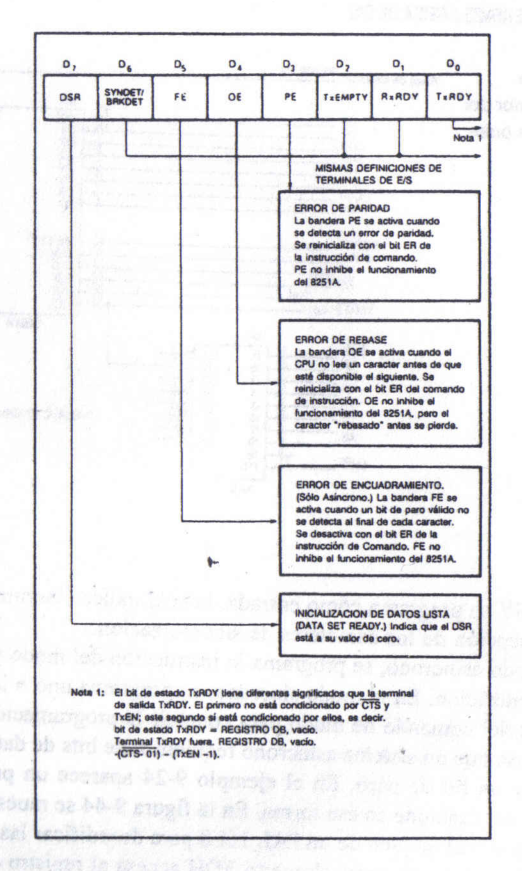


FIGURA 9-45 Palabra de comando del 8251A. (Cortesía de Intel Corporation.)

FIGURA 9–46 La palabra de estado de la 8251A. (Cortesía de Intel Corporation.)



Antes que sea posible escribir el programa para enviar o recibir datos seriales con el 8251A, necesitamos conocer la función del registro de estado (figura 9-46). El registro de estado contiene información acerca de condiciones de error y los estados de DSR, SY/BD, T×EMPTY, R×RDY y T×RDY.

Supóngase que se escribe un procedimiento (véase Ejemplo 9-26) para transmitir el contenido de AH a través de la terminal de salida (T×D) de datos seriales del 8251A. El programa examina continuamente (encuesta) el bit de T×RDY, para determinar si el transmisor está listo para recibir datos. En ese procedimiento se utiliza el circuito de la figura 9-44.

EJEMPLO 9–26

```
                ;Procedimiento que transmite el contenido de AH
                ;
= 00FF          CNTR    EQU    0FFH           ;puerto para comando
= 00FD          DATA    EQU    0FDH           ;puerto para datos
```

```
0000                SEND     PROC    NEAR

0000 50                      PUSH AX                    ;salvar AX
                    ;probar TxRDY

0001                SEND1:

0001 E4 FF                   IN   AL,CNTR               ;leer estado
0003 D0 C8                   ROR  AL,1                  ;rotar TxRDY a CF
0005 73 FA                   JNC  SEND1                 ;si no está listo

                    ;transmitir datos

0007 8A C4                   MOV  AL,AH
0009 E6 FD                   OUT  DATA,AL

000B 58                      POP  AX                    ;restaurar AX
000C C3                      RET

000D                SEND     ENDP
```

Para leer la información recibida por el 8251A, se prueba el bit R×RDY del registro de estado. En el ejemplo 9-27 se presenta un procedimiento para probar ese bit y decidir si el 8251A ha recibido los datos. Al ocurrir la recepción de datos, el procedimiento hace la prueba de errores. Si se detecta un error, el procedimiento retorna con AL =?. Si no ha ocurrido error, entonces el procedimiento retorna con AL igual al carácter recibido.

EJEMPLO 9–27

```
                    ;Procedimiento que recibe datos de la 8251A
                    ;
= 00FF              CNTR     EQU    0FFH            ;puerto para comando
= 00FD              DATA     EQU    0FDH            ;puerto para datos
= 0004              MASKS    EQU    4               ;máscara RXRDY
= 0038              ERROR    EQU    38H             ;máscara de error
= 0015              E_Res    EQU    15H             ;reinicializar error
= 003F              QUES     EQU    '?'             ;signo de interrogación

0000                RECV     PROC   NEAR

                    ;probar RXRDY

0000                RECV1:

0000 E4 FF                   IN   AL,CNTR           ;leer estado
0002 A8 04                   TEST AL,MASKS          ;probar RXRDY
0004 74 FA                   JZ   RECV1             ;si no está listo

                    ;probar si hay errores

0006 A8 38                   TEST AL,ERROR
0008 75 03                   JNZ  ERR               ;si hay un error

                    ;leer datos
```

```
000A E4 FD              IN   AL,DATA
000C C3                 RET

             ;error

000D         ERR:

000D B0 15              MOV  AL,E_RES          ;reinicializar error
000F E6 FF              OUT  CNTR,AL
0011 B0 3F              MOV  AL,QUES           ;obtener interrogación (?)
0013 C3                 RET

0014         RECV       ENDP
```

Los tipos de errores detectados por el 8251A son errores de paridad, error de encuadre y error de rebase. Un error de paridad indica que los datos recibidos tienen una paridad incorrecta. Un error de encuadramiento indica que los bits de arranque y paro no están en su lugar correcto. Un error de rebase indica que los datos se han "desbordado" del registro interno del receptor. Estos errores no deberían ocurrir durante el funcionamiento normal. Si ocurre un error de paridad, un error de encuadramiento ocurre si el receptor recibe datos a una frecuencia en baudios incorrecta. Un error de desbordamiento sólo ocurre si el programa no lee los datos del USART.

9–7 CONVERTIDORES ANALOGICO/DIGITAL (ADC) Y DIGITAL/ANALOGICO (DAC)

Los convertidores analógico/digital (ADC) y digital/analógico (DAC) se utilizan para conectar el microprocesador con el mundo analógico. Muchos de los acontecimientos que se monitorean y controlan con el microprocesador, son analógicos. A menudo incluyen vigilancia de todas las formas de acontecimientos, incluso voz, hasta el control de motores y dispositivos similares. Para poder efectuar la interface del microprocesador con estos acontecimientos, se deben conocer la interface y el control del ADC y el DAC que convierten datos analógicos a digitales y viceversa.

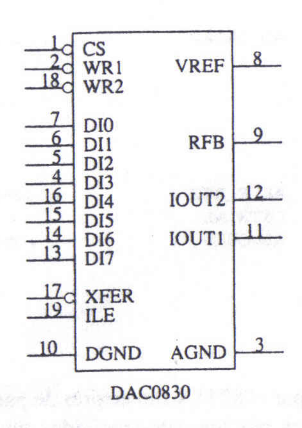
Convertidor DAC0830 de digital a analógico

El DAC0830[2] es un convertidor de digital a analógico bastante común y de bajo costo. Es un convertidor de 8 bits que transforma un número binario de 8 bits a un voltaje analógico. Hay disponibles otros convertidores que convierten 10, 12 o 16 bits a voltajes analógicos. El número de escalones de voltaje generados por el convertidor es igual al número de combinaciones binarias de entrada. Por tanto, un convertidor de 8 bits genera 256 niveles diferentes de voltaje, un convertidor de 10 bits genera 1 024 niveles de voltaje, etc. El DAC0830 es de mediana velocidad y transforma una entrada digital a una salida analógica en alrededor de 1.0 microsegundos.

En la figura 9-47 se ilustra el diagrama de base del DAC0830, el cual tiene un conjunto de ocho conexiones para un canal de datos para la aplicación del código digital de entrada y un par

[2]The DAC0830 es un producto National Semiconductor Corporation.

FIGURA 9–47 Diagrama de base de convertidor digital/analógico DAC0830.



de salidas analógicas, etiquetas Iout1 y Iout2, que se emplean como salidas a un amplificador operacional externo. Debido a que es un convertidor de 8 bits, su "escalón" de voltaje de salida se define como –Vref (voltaje de referencia) dividido entre 255. Por ejemplo, si el voltaje de referencia es de –5.0 V, cada "escalón" de voltaje de salida es de +.0196078431373 volts lo cual define la resolución del convertidor. Se debe tener en cuenta que el voltaje de salida tiene polaridad opuesta a la del voltaje se referencia. Si se aplica una entrada de 1001 0010 al convertidor, el voltaje de salida será el voltaje escalonado multiplicado por 1001 0010 o, en este caso, de +2.86274509804 volts. Si se cambia el voltaje de referencia a –5.1 volts, el escalón de voltaje escalonado se vuelve +.02 V.

Estructura interna del DAC0830. En la figura 9-48 se ilustra la estructura interna del DAC0830. Se verá que contiene dos registros internos. El primero, es un registro de "retención" y el segundo se conecta con el convertidor de escalera interno R-2R. Los dos registros permiten retener un byte mientras se convierte otro. En muchos casos, se deshabilita el primer registro y se utiliza sólo el segundo para la entrada de datos al convertidor. Para lograrlo, se lleva a un 1 lógico la terminal ILE y a un 0 lógico la terminal $\overline{CS}$ (selección de integrado).

Ambos registros internos del DAC0830 son transparentes. Cuando la entrada G del registro es un 1 lógico, los datos pasan por él, pero cuando la entrada G es un 0 lógico, entonces se retienen los datos. El convertidor tiene una terminal de entrada para un voltaje de referencia (Vref) que establece el voltaje de salida total. Si se aplican –10V en Vref, el voltaje de salida (1111 1111) total es de +10V. La salida de la red de escalera R-2R del convertidor funciona con Iout1 y Iout2. Estas salidas están diseñadas para aplicarlas a un amplificador operacional como el 741 o equivalente.

Conexión del DAC0830 con el microprocesador. El DAC0830 se conecta con el microprocesador como se ilustra en la figura 8-49. Se utiliza un PAL 16L8 para decodificar el DAC0830 en una dirección 20H de puerto de E/S de 8 bits. Siempre que se ejecuta una instrucción OUT 20H,AL, el contenido del canal de datos (AD0 –AD7) se pasa al convertidor. El amplificador operacional 741 junto con el voltaje Zener de referencia de –12V ocasiona que el voltaje de salida total sea de +12V. El amplificador operacional alimenta al manejador de un motor de CD de 12 volts. Este manejador es un amplificador Darlington para motores grandes. En este ejemplo se muestra que el convertidor maneja un motor, pero se podrían utilizar otros dispositivos como salida.

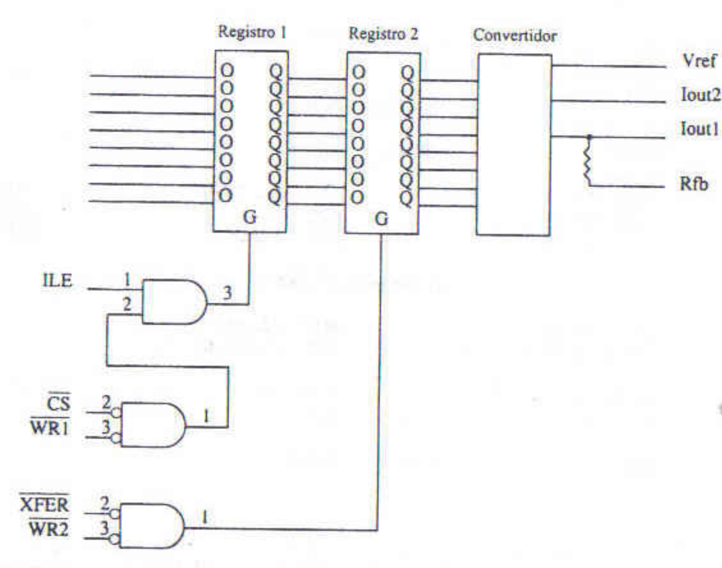


FIGURA 9-48 La estructura interna del DAC0830.

El convertidor ADC080X de analógico a digital

Un ADC común de bajo costo, es del ADC0804 que pertenece a una familia de convertidores que son casi iguales excepto en la exactitud. Este convertidor es compatible con una amplia gama de microprocesadores, como el 8086. Hay disponibles ADC más rápidos y algunos con mayor resolución que 8 bits, pero este convertidor es ideal para muchas aplicaciones que no requieren un alto grado de exactitud. El ADC0804 requiere hasta 100 microsegundos para convertir una entrada analógica de voltaje a una salida en código digital.

En la figura 9-50 se ilustran las conexiones de terminales del convertidor ADC0804[3]. Para que funcione, se "pulsa" la terminal $\overline{WR}$ con $\overline{CS}$ conectada a tierra para iniciar el funcionamiento del convertidor. Debido a que este convertidor requiere una cantidad considerable de tiempo para la conversión, una terminal etiquetada INTR señala el fin de la conversión. En la figura 9-51 aparece un diagrama de temporización que muestra la interacción de las señales de control. Como se puede ver, se pone a funcionar el convertidor con el pulso $\overline{WR}$, se espera hasta que INTR sea un 0 lógico, luego, se leen los datos del convertidor. Si se utiliza un retardo de tiempo que permita, cuando menos, 100 microsegundos, entonces no se necesita probar la terminal INTR. Otra opción es conectar la terminal INTR con una entrada de interrupción, a fin de que cuando concluya la conversión, ocurra una interrupción.

La señal analógica de entrada. Antes de poder conectar el ADC0804 con el microprocesador, se deben entender sus entradas analógicas. Hay dos entradas analógicas a este convertidor, Vin (+) y Vin (–). Estas entradas están conectadas con un amplificador operacional interno y son entradas

---

[3]The ADC0804 es un producto de National Semiconductor Corporation.

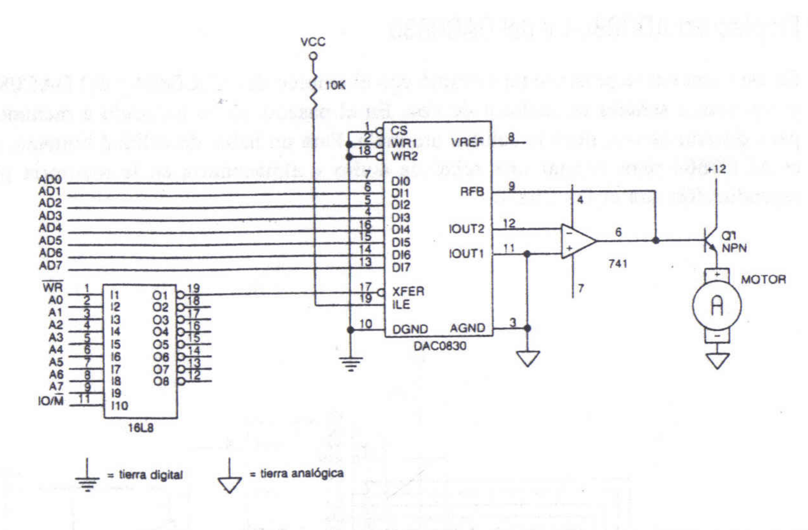
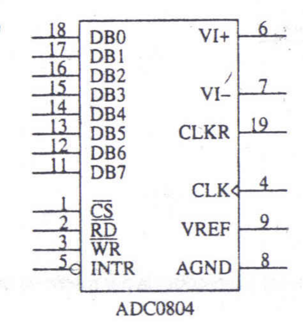


FIGURA 9–49 Un DAC0830 en interface con el microprocesador 8086 en la localidad 20H de E/S de 8 bits.

diferentes, como se ilustra en la figura 9-52. Las entradas diferenciales se suman en el amplificador operacional para producir una señal para el convertidor analógico/digital interno. En la figura 9-52 se ilustran algunas formas para utilizar estas entradas diferenciales. En la primera forma (figura 9-52a) se emplea una sola entrada que puede variar entre cero y +5.0 volts. La segunda (figura 9-52b) muestra un voltaje variable aplicado a la terminal Vin (–) a fin de poder ajustar una referencia de cero para VIN (+).

Generación de la señal de reloj. El ADC0804 requiere un reloj para funcionar. El reloj puede ser externo, conectado con la terminal CLK IN o se puede generar con un circuito de RC. El intervalo permisible para las frecuencias de reloj es entre 100 KHz y 1460 KHz. Es deseable utilizar una frecuencia lo más cercana posible a 1460 KHz para que el tiempo de conversión sea mínimo.

FIGURA 9–50 Diagrama de base del convertidor analógico/digital ADC0804.



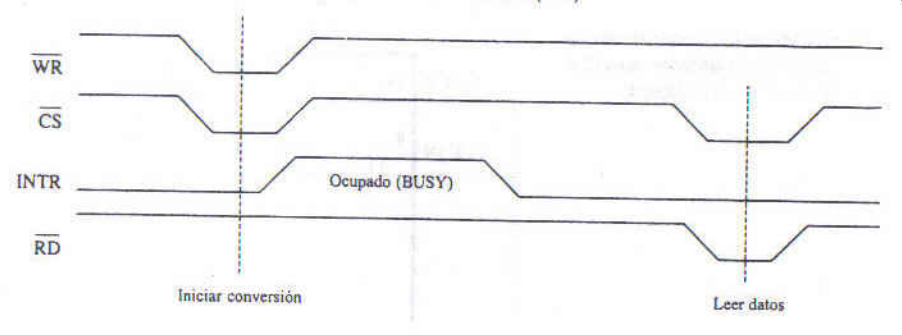


FIGURA 9-51 La temporización para el convertidor analógico/digital ADC0804.

Si el reloj se genera con un circuito RC, se utilizan las terminales CLK IN y CLK R conectadas con un circuito RC, como se ilustra en la figura 9-53. Cuando se utiliza esta conexión, la frecuencia de reloj se calcula con la siguiente ecuación:

$$\text{Fclk} = \frac{1}{1.1\,RC}$$

Conexión del ADC0804 con el microprocesador. El ADC0804 en interface con el microprocesador 8088, como se ilustra en la figura 9-54. Se verá que Vref no está conectado, lo cual es normal en algunos casos. Supóngase que el ADC0804 se decodifica en el puerto E/S de 8 bits, en la dirección 40H para los datos y en la dirección 42H para la señal INTR y que se requiere un procedimiento para poner en marcha y leer los datos del ADC. Este procedimiento se presenta en el ejemplo 9-28. Se debe tener en cuenta que se "encuesta" el bit INTR y cuando es un 0 lógico, el procedimiento termina con AL conteniendo el código digital de la conversión.

FIGURA 9-52 Entradas analógicas del convertidor ADC0804. (a) Para detectar una entrada de 0 a +5.0 volts. (b) Para detectar una entrada con un desvío respecto a tierra.

Entrada analógica
Vin (+)
Vin (−)
(a)

Entrada analógica
Vin (+)
Vin (−)
+5V
(b)

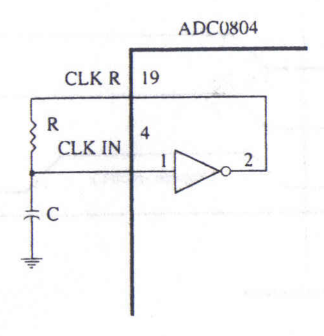


FIGURA 9–53 Conexión del circuito RC con las terminales CLK IN y CLK R del ADC0804.

EJEMPLO 9–28

```
                        ;Procedimiento que lee datos del ADC y retorna
                        ;con ellos en AH
                        ;
0000                    ADCX      PROC    NEAR

0000 E6 40                        OUT  40H,AL          ;iniciar conversión

0002                    ADCX1:

0002 E4 42                        IN   AL,42H          ;leer INTR
0004 A8 80                        TEST AL,80H          ;probar INTR
0006 75 FA                        JNZ  ADCX1           ;repetir hasta que INTR = 0

0008 E4 40                        IN   AL,40H          ;obtener datos del ADC
000A C3                           RET

000B                    ADCX      ENDP
```

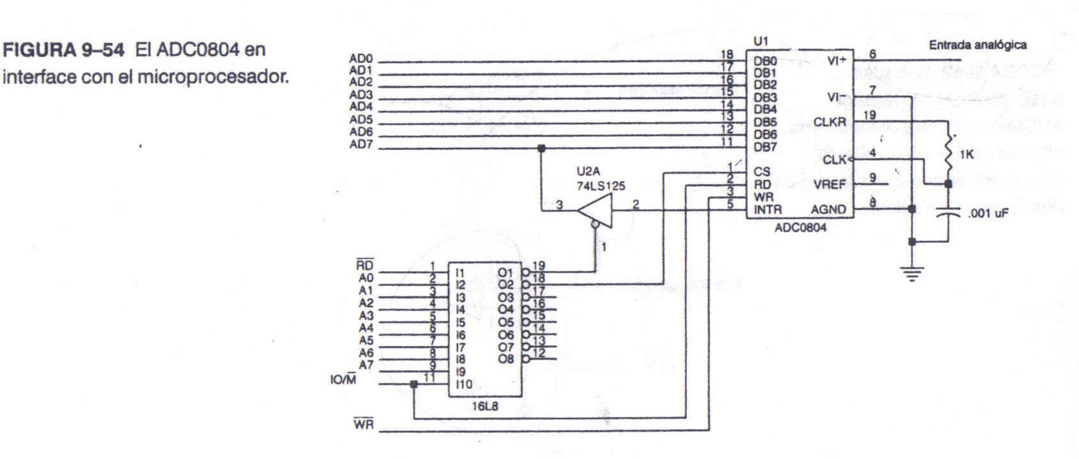


FIGURA 9–54 El ADC0804 en interface con el microprocesador.

Empleo del ADC0804 y del DAC0830

En esta sección se presenta un ejemplo con el empleo del ADC0804 y del DAC0830 para capturar y reproducir señales de audio o de voz. En el pasado, se ha utilizado a menudo un sintetizador para generar la voz, pero la calidad era mala. Para un habla de calidad humana, se puede utilizar el ADC0804 para atrapar una señal de audio y almacenarla en la memoria para su posterior reproducción con el DAC0830.

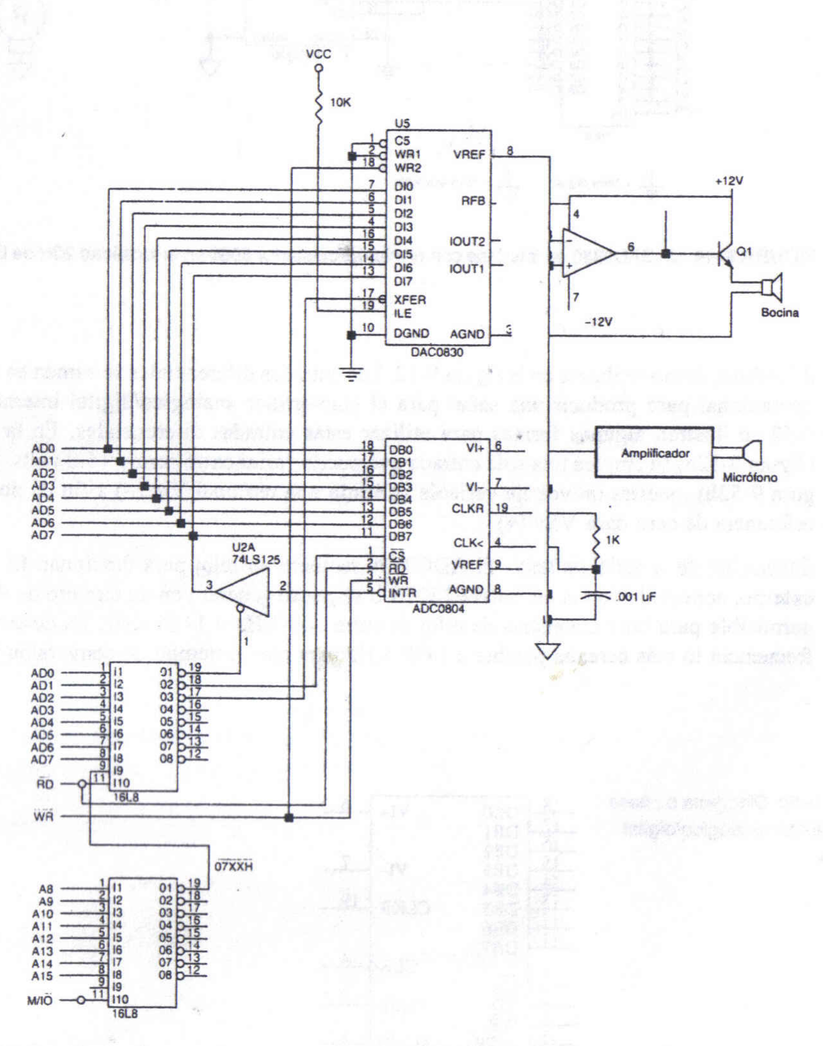


FIGURA 9–55 Un circuito para almacenar (voz) y reproducirla por medio de una bocina.

En la figura 9-55 se ilustran los circuitos requeridos para conectar el ADC0804 en los puertos 0700H y 0702H de E/S de 16 bits. El DAC0830 está en interface en el puerto 704H de E/S. El programa que hace funcionar estos convertidores aparece en el ejemplo 9-29. Este programa lee una ráfaga de voz durante un segundo y la reproduce 10 veces. Este proceso se repite hasta que se apague el sistema.

EJEMPLO 9–29

```
                        ;Programa que registra un paso de 1 segundo de voz
                        ;y lo reproduce 10 veces antes de registrar
                        ;el siguiente segundo de habla
                        ;
                        ;Se supone que el reloj es de 8 MHz
                        ;
0000                    STAC    SEGMENT STACK

0000 0800 [                     DW   2048 DUP (?)          ;espacio en la pila
          0000
                ]
1000                    STAC    ENDS

0000                    DATA1   SEGMENT 'DATA'

0000 03E8 [             WORDS   DB   1000 DUP (?)          ;espacio para el siguiente segundo de habla
          00
             ]
03E8                    DATA1   ENDS

0000                    CODE1   SEGMENT 'CODE'

                                ASSUME  CS:CODE1,DS:DATA1,SS:STAC

0000                    START   PROC   FAR

0000 B8 ---- R                  MOV  AX,DATA1              ;cargar DS
0003 8E D8                      MOV  DS,AX
0005 33 C0                      XOR  AX,AX                 ;cargar ES
0007 8E C0                      MOV  ES,AX

0009 E8 000A                    CALL  READ                 ;leer voz

000C B9 000A                    MOV  CX,10

000F                    START1:

000F E8 0023                    CALL WRITE                 ;reproducir voz
0012 E2 FB                      LOOP START1                ;repetir 10 veces

0014 EB EA                      JMP  START                 ;rehacer todo

0016                    START   ENDP

0016                READ        PROC    NEAR

0016 BF 0000 R                  MOV  DI,OFFSET WORDS       ;direccionar datos
0019 B9 03E8                    MOV  CX,1000               ;cargar conteo
001C BA 0700                    MOV  DX,0700H              ;direccionar puerto
```

```
001F                 READ1:

001F EE                         OUT  DX,AL          ;arrancar el convertidor
0020 83 C2 02                   ADD  DX,2

0023                 READ2:

0023 EC                         IN   AL,DX          ;obtener INTR
0024 A8 80                      TEST AL,80H         ;probar INTR
0026 75 FB                      JNZ  READ2          ;esperar a que INTR = 0

0028 83 EA 02                   SUB  DX,2
002B EC                         IN   AL,DX          ;obtener los datos
002C 88 05                      MOV  [DI],AL        ;salvar los datos
002E 47                         INC  DI

002F E8 0018                    CALL DELAY          ;esperar un milisegundo (ms)

0032 E2 EB                      LOOP READ1          ;repetir 1024 veces
0034 C3                         RET

0035                 READ       ENDP

0035                 WRITE      PROC   NEAR

0035 51                         PUSH CX
0036 BF 0000 R                  MOV  DI,OFFSET WORDS ;direccionar datos
0039 B9 03E8                    MOV  CX,1000        ;cargar conteo
003C BA 0704                    MOV  DX,0704H       ;direccionar puerto
003F                 WRITE1:

003F 8A 05                      MOV  AL,[DI]        ;obtener datos
0041 EE                         OUT  DX,AL          ;dar salida a los datos
0042 47                         INC  DI

0043 E8 0004                    CALL DELAY          ;esperar 1 milisegundo (ms)

0046 E2 F7                      LOOP WRITE1
0048 59                         POP  CX
0049 C3                         RET

004A                 WRITE      ENDP

004A                 DELAY      PROC   NEAR

                ;wait 1/1000 second

004A 51                         PUSH CX
004B B9 03E8                    MOV  CX,1000

004E                 DELAY1:

004E E2 FE                      LOOP DELAY1
0050 59                         POP  CX
0051 C3                         RET

0052                 DELAY      ENDP

0052                 CODE1      ENDS

                                END    START
```

9–8 RESUMEN

1. El 8086 tiene dos tipos básicos de instrucciones de E/S: IN y OUT. La instrucción IN da entrada a los datos desde un dispositivo externo de E/S hacia el registro AL (8 bits) o el AX (16 bits). La instrucción IN está disponible como instrucción para puerto fijo, para puerto variable o como instrucción en cadena (80286–80486), INSB o INSW. La instrucción OUT da salida a los datos de AL o de AX a un dispositivo de E/S externo y también está disponible como instrucción fija, variable o en cadena OUTSB o OUTSW. En la instrucción para puerto fijo se utiliza una dirección de puerto de E/S de 8 bits; en las instrucciones variables y para cadenas se utiliza un número de puerto de 16 bits que se encuentra en el registro DX.
2. En la E/S aislada, llamada a veces E/S directa, se emplea un mapa separado para el espacio de E/S lo cual libera toda la memoria para que la utilice el programa. En la E/S aislada se utilizan las instrucciones IN y OUT para transferir datos entre el dispositivo de E/S y el microprocesador. La estructura de control del mapa de E/S utiliza las señales $\overline{IORC}$ (control de lectura de E/S) y $\overline{IOWC}$ (control de escritura en E/S), así como las señales de selección de banco $\overline{BHE}$ y A0 para efectuar una transferencia de E/S. El 8086/8088 empleaban la señal $M/\overline{IO}$ ($IO/\overline{M}$) para generar las señales de control de E/S.
3. El espacio de E/S mapeado en la memoria emplea una parte del espacio de la memoria para las transferencias de E/S. Con ello se reduce el espacio de la memoria, pero se evita la necesidad de emplear las señales $\overline{IORC}$ y $\overline{IOWC}$ para las transferencias de E/S. Además, cualquier instrucción que direccione la memoria con cualquier modo de direccionamiento se puede emplear para transferir datos entre el microprocesador y el dispositivo de E/S con el empleo del espacio de E/S mapeado en la memoria.
4. Todos los dispositivos de entrada tienen registros, de modo que los datos de E/S sólo se conecten al canal de datos durante la ejecución de la instrucción IN. Los registros están integrados en un periférico programable o aparte.
5. En todos los dispositivos de salida se utiliza un registro transparente para retener los datos de salida durante la ejecución de la instrucción OUT. Esto es necesario porque los datos aparecen en el canal de datos por menos de 100 ns para una instrucción OUT y la mayor parte de los dispositivos de salida necesitan los datos mayor tiempo. En muchos casos, el registro transparente está integrado en el periférico.
6. El reconocimiento (handshaking) o encuesta es la acción de dos dispositivos independientes que se sincronizan con algunas líneas de control. Por ejemplo, la computadora le pregunta a la impresora si está ocupada consultando a la señal BUSY de la impresora. Si no está ocupada, la computadora da salida a los datos a la impresora y le informa a la impresora que los datos están disponibles mediante una señal de habilitación (DS) de datos. Esta comunicación entre la computadora y la impresora se llama reconocimiento (handshake).
7. El número del puerto de E/S aparece en las conexiones A7–A0 del canal de direcciones para una instrucción con puerto de E/S fijo y en A15–A0 para una instrucción con puerto E/S variable. En ambos casos, los bits de dirección por arriba de A15 son cero. La dirección de E/S de 8 bits que se encuentra en A7–A0, también contiene ceros lógicos en las conexiones A15–A8 de dirección.
8. Debido a que los 8086, 80286 y 80386SX tienen un canal de datos de 16 bits y la E/S direcciona a localidades de tamaño de byte, el espacio de E/S también está organizado en bancos como

el sistema de memoria. Para poder conectar un dispositivo de E/S de 8 bits con el canal de datos de 16 bits, a menudo se requieren señales de habilitación estroboscópica separadas de escritura, una superior y una inferior, para las operaciones de escritura en el espacio de E/S.

9. El decodificador de puertos de E/S es muy similar al de direcciones de memoria, excepto que en lugar de decodificar toda la dirección, el decodificador de puerto de E/S sólo decodifica una dirección de 16 bits para instrucciones de puerto variable y, a menudo, un número de puerto de 8 bits, para instrucciones de E/S de puerto fijo.
10. El 8255 es una interface periférica programable que tiene 24 terminales de E/S, programables en dos grupos de 12 terminales cada uno (grupo A y grupo B). Funciona en tres modos: E/S simple (modo 0), E/S con señales de habilitación estroboscópica (modo 1) y E/S bidireccional (modo 2). Cuando el 8255 se conecta con un 8086 que trabaje a 8 MHz, se insertan dos estados de espera porque la velocidad del microprocesador es mayor de la que puede manejar el 8255.
11. El 8279 es un controlador programable de teclado/exhibición visual, que puede controlar un teclado de 64 teclas y una exhibición visual numérica de 16 dígitos.
12. El 8254 es un temporizador programable de intervalos que tiene tres contadores de 16 bits, que cuentan en binario o en decimal en código binario (BCD). Cada contador es independiente de los otros y funciona en seis modos diferentes. Los seis modos del contador son: (1) contador de eventos; (2) multivibrador monoestable que se puede volver a disparar; (3) generador de pulsos; (4) generador de onda cuadrada; (5) generador de pulsos disparado por programa; (6) generador de pulsos disparado por señal.
13. El 8251A es una interface programable para comunicaciones que puede recibir o transmitir datos seriales en forma asíncrona y síncrona.
14. El DAC0830 es un convertidor de digital/analógico de 8 bits que convierte una señal digital a un voltaje analógico en un tiempo de 1 microsegundo.
15. El ADC0804 es un convertidor analógico/digital de 8 bits que convierte una señal analógica a una señal digital en un tiempo de 100 microsegundos.

9–9 PREGUNTAS Y PROBLEMAS

1. Explique en cuál sentido fluyen los datos con una instrucción IN y con una OUT.
2. ¿Dónde se almacena el número de puerto de E/S para una instrucción E/S fija?
3. ¿Dónde se almacena el número de puerto de E/S para una instrucción E/S variable?
4. ¿Dónde se almacena el número de puerto de E/S para una instrucción de E/S en cadena?
5. ¿A cuál registro se da entrada a los datos con la instrucción IN de 16 bits?
6. Describa el funcionamiento de la instrucción OUTSB.
7. Describa el funcionamiento de la instrucción INSW.
8. Compare un sistema de E/S incluido en la memoria con un sistema E/S aislado.
9. ¿Qué es la interface básica de entrada?
10. ¿Qué es la interface básica de salida?
11. Explique el término reconocimiento (handshaking) según se aplica a los sistemas de E/S de computadora.

12. Una dirección de puerto de E/S con número par, se encuentra en el banco __________ de E/S en el microprocesador 8086.
13. Muestre los circuitos requeridos para generar las señales de habilitación estroboscópica superior e inferior para escritura en E/S.
14. Desarrolle un decodificador de puerto de E/S con el empleo de un 74ALS138 que genere señales de habilitación estroboscópica de E/S para un banco bajo, para las direcciones de puerto de E/S: 10H, 12H, 14H, 16H, 18AH, 1CH y 1EH de 8 bits.
15. Desarrolle un decodificador de puerto de E/S con el empleo de un 74ALS138 que genere señales de habilitación estroboscópica de E/S para el banco alto, para las direcciones de puerto de E/S: 11H, 13H, 15H, 17H, 19H, 1BH, 1DH y 1FH de 8 bits.
16. Desarrolle un decodificador de puerto de E/S con el empleo del PAL16L8, que genere 16 señales de habilitación estroboscópica de E/S para las 16 direcciones de puerto: 1000H–1001H, 1002H–1003H, 1004H–1005H, 1006H–1007H, 1008H–1009H, 100AH–100BH, 100CH–100DH y 100EH–100FH.
17. Desarrolle un decodificador de puerto de E/S con el empleo del PAL16L8 que genere las siguientes señales de habilitación estroboscópica para el banco bajo: 00A8H, 00B6H,y 00EEH.
18. Desarrolle un decodificador de puerto de E/S con el empleo del PAL16L8, que genere las siguientes señales de habilitación estroboscópica para el banco alto: 300DH, 300BH, 1005H y 1007H.
19. ¿Por qué se ignoran $\overline{BHE}$ y A0 en un decodificador de dirección de puerto de 16 bits?
20. Un dispositivo de E/S de 8 bits, ubicado en la dirección 0010H de puerto de E/S, ¿con cuáles conexiones del canal de datos se conecta?
21. Un dispositivo de E/S de 8 bits, ubicado en la dirección 100DH de puerto de E/S, ¿con cuáles conexiones del canal de datos se conecta?
22. ¿Cuántas terminales de E/S programables tiene el 8255?
23. Enumere las terminales que pertenecen al grupo A y al grupo B en el 8255.
24. ¿Cuáles dos terminales del 8255 logran la selección de la dirección del puerto de E/S interno para el 8255?
25 La conexión $\overline{RD}$ en el 8255, ¿con cuál conexión de canal de control del sistema de 8086 se conecta?
26. Con el empleo de un PAL16L8 efectúe la interface de un 8255 con el microprocesador 8086 de modo que funcione en las ubicaciones 0380H, 0382H, 0384H y 0386H de E/S.
27. Cuando se reinicializa el 8255 se inicializa a todos los puertos de E/S como__________.
28. ¿Cuáles tres modos de funcionamiento están disponibles para el 8255?
29. ¿Cuál es la finalidad de la señal $\overline{STB}$ en una operación de entrada por habilitación al 8255?
30. Explique el funcionamiento de un motor de pasos sencillo de 4 devanados.
31. ¿Qué activa a la terminal IBF en una operación de habilitación de entrada por señal estroboscópica al 8255?
32. Redacte un programa para hacer 1 lógico la terminal PC7 del 8255 durante el funcionamiento de habilitación de entrada por señal estroboscópica.
33. ¿Cómo se habilita la terminal de solicitud de interrupción (INTR) en el modo de entrada por habilitación al 8255?
34. En el funcionamiento de salida por habilitación del 8255, ¿cuál es la finalidad de la señal $\overline{ACK}$?
35. ¿Qué desactiva la señal $\overline{OBF}$ en el funcionamiento por habilitación de salida del 8255?

36. Redacte el programa requerido para saber si PC4 es un 1 lógico cuando se hace funcionar al 8255 en el modo de salida por habilitación.
37. ¿Qué grupo de terminales se utiliza durante el funcionamiento bidireccional del 8255?
38. ¿Cuáles terminales son E/S de uso general durante el funcionamiento del 8255 en el modo 2?
39. ¿Qué se conecta con la terminal CLK del 8279?
40. ¿Cuántos estados de espera se requieren para la interface del 8279 con el microprocesador 8086 trabajando con un reloj de 8 MHz?
41. Si la terminal CLK del 8279 se conecta con un reloj de 3.0 MHz, programe el reloj interno.
42. ¿Qué es un error de sobreflujo en el 8279?
43. ¿Cuál es la diferencia entre codificado y decodificado según se definen para el 8279?
44. Haga la interface del 8279 de modo que funcione como puertos 40H–7FH de E/S de 8 bits. Utilice el decodificador 74ALS138 y utilice el canal de datos superior o inferior.
45. Haga la interface de un teclado de 16 teclas y una exhibición visual numérica de 8 dígitos con el 8279.
46. El temporizador 8254 de intervalos funciona desde DC hasta _____Hz.
47. ¿En cuántos modos diferentes funciona cada contador del 8254?
48. Haga la interface de un 8254 para que funcione en las direcciones de puerto de E/S XX10H, XX12H, XX14H y XX16H. Escriba el programa requerido para hacer que el contador 2 genere una onda cuadrada de 80 KHz si la entrada CLK al contador 2 es de 8 MHz.
49. ¿Qué número se programa en un contador 8254 para que cuente 300 eventos?
50. Si se programa un conteo de 16 bits en el 8254, ¿cuál byte del conteo se programa primero?
51. Explique cómo funciona la palabra de retrolectura en el 8254.
52. Programe el contador 1 del 8254 para que genere una serie continua de pulsos que estén en alto 10 microsegundos y en bajo 1 microsegundo. No olvide indicar la frecuencia CLK requerida para lograr esta tarea.
53. ¿Por qué un ciclo de trabajo de 50% hace que el motor se quede parado en el circuito de control de velocidad y sentido de rotación del motor presentado en este capítulo?
54. ¿Qué son datos seriales asíncronos?
55. ¿Qué son datos seriales síncronos?
56. ¿Qué es la frecuencia en baudios?
57. Programe el 8251A para funcionamiento síncrono con el empleo de 6 bits de datos, paridad par, un bit de paro y un divisor de 1 de la frecuencia en baudios. (Supóngase que los puertos de E/S están numerados 20H–22H.)
58. Si el 8251A ha de generar una señal en serie síncrona con una frecuencia de 2400 baudios y si el divisor está programado para 16, ¿cuál es la frecuencia de la señal conectada con la terminal T×C?
59. Describa los siguientes términos: símplex, semidúplex y dúplex total.
60. ¿Cómo se reinicializa el 8251A?
61. El DAC0830 convierte una entrada digital de 8 bits a una salida analógica más o menos en cuánto tiempo________?
62. ¿Cuál es la resolución salida del DAC0830 si el voltaje de referencia es de –2.55 V?
63. Efectúe la interface de un DAC0830 para que funcione en el puerto 400H de E/S.
64. Programe la interface de la pregunta 62 de modo que el DAC0830 genere una forma de onda triangular de voltaje. La frecuencia aproximada de esa forma de onda debe ser de 100 Hz.
65. El ADC080X requiere un tiempo de alrededor de ________, para convertir un voltaje analógico a un código digital.

66. ¿Cuál es la finalidad de la terminal INTR en el ADC080X?
67. ¿Para qué finalidad se utiliza la terminal $\overline{WR}$ en el ADC080X?
68. Efectúe la interface de un ADC080X en el puerto 0260H de E/S para datos y en el 0270H para probar la terminal INTR.
69. Desarrolle un programa para el ADC080X en la pregunta 68, para que lea el voltaje de entrada cada 100 ms y que almacene el resultado en un arreglo de memoria de 100H bytes de longitud.

CAPITULO 10

Interrupciones

INTRODUCCION

En este capítulo se amplía la cobertura de las interfaces periféricas básicas de E/S programables, con el examen de una técnica llamada E/S procesada por interrupción. Una interrupción es un procedimiento generado por un periférico, que interrumpe cualquier programa que se corra en ese momento.

En este capítulo se presentan ejemplos y una explicación detallada de la estructura de interrupción en toda la línea de microprocesadores Intel.

OBJETIVOS DEL CAPITULO

Al concluir este capítulo, usted podrá:

1. Explicar la estructura de interrupción de los microprocesadores de Intel.
2. Explicar el funcionamiento de las instrucciones de interrupción que son: INT, INTO, INT3 y BOUND.
3. Explicar la forma en que el bit (TF) de bandera de interrupción modifica la estructura de interrupción.
4. Describir la función del bit (TF) de trampa y el procedimiento de trazado o seguimiento generado por este bit.
5. Establecer procedimientos de servicio de interrupción que controlen los periféricos externos de baja velocidad.
6. Ampliar la estructura de interrupción del microprocesador con el controlador programable de interrupciones 8259A y otras técnicas.
7. Explicar la finalidad y funcionamiento de un reloj de tiempo real.

10–1 PROCESAMIENTO BASICO DE LAS INTERRUPCIONES

En esta sección se describe la función de una interrupción en un sistema basado en un microprocesador así como la estructura y características de las interrupciones disponibles en los microprocesadores Intel.

La finalidad de las interrupciones

Las interrupciones son de particular utilidad cuando se conectan (interface) dispositivos de E/S que suministran o requieren datos a velocidades de transferencia más o menos bajas. En el capítulo 9, por ejemplo, se vio el ejemplo de un teclado con el empleo del funcionamiento con entrada de habilitación estroboscópica del 8255. En ese ejemplo, el programa examinó el 8255 y su bit IBF para saber si había datos disponibles desde el teclado. Si el usuario del teclado tecleaba un caracter por segundo, el programa del 8255 esperaba un segundo completo entre cada pulsación para que la persona oprimiera otra tecla. Este método es una pérdida tan enorme de tiempo, que los diseñadores han creado otro proceso llamado procesamiento por interrupciones para manejar esta situación.

Al contrario de la técnica de encuesta, el procesamiento de interrupciones permite al microprocesador ejecutar otro programa mientras el operador piensa qué tecla va a oprimir apenas se oprime la tecla, el decodificador del teclado elimina los "rebotes" y da salida a un pulso que interrumpe al microprocesador. En esta forma, el microprocesador ejecuta otro programa hasta el momento en que se oprime la tecla y, entonces, la lee y retorna al programa que interrumpió. Como resultado, el microprocesador puede imprimir informes o concluir cualquier otra tarea mientas el usuario teclea un documento y piensa en lo que tecleará después.

En la figura 10-1 se ilustra una línea de tiempo que indica un usuario que teclea datos en un teclado, una impresora que lee datos de la memoria y la ejecución de un programa.

El programa es el programa principal que se interrumpe con cada tecla oprimida y con cada carácter que va a imprimir la impresora. Se debe tener en cuenta que el procedimiento de servicio de interrupción del teclado, llamado por la interrupción del teclado y el procedimiento de servicio de interrupción de la impresora requieren poco tiempo para su ejecución.

Interrupciones

Las interrupciones en toda la línea de microprocesadores Intel incluyen dos terminales a través de las que solicitan interrupciones (INTR Y NMI) y una terminal ($\overline{INTA}$) que "reconoce" la interrup-

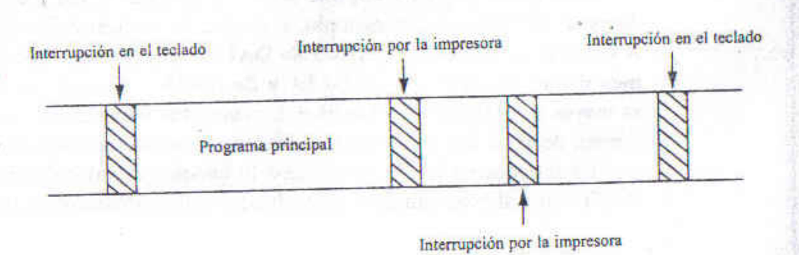


FIGURA 10–1 Una línea de tiempo que indica el empleo de las interrupciones en un sistema típico.

ción solicitada a través de INTR. Además de esas terminales, el microprocesador también tiene las instrucciones para interrupciones INT, INTO, INT3 y BOUND. También se utilizan dos bits del registro de banderas IF (bandera de interrupción) y TF (bandera de trampa) en la estructura de interrupción y también una instrucción especial para retorno IRET (o IRETD en el 80386 o el 80486).

Vectores de interrupción. Los vectores de interrupción y la tabla de estos vectores son de máxima importancia para entender las instrucciones por hardware y por software. La *tabla de vectores de interrupción* se encuentra en los primeros 1024 bytes de memoria en las direcciones 000000H hasta 0003FFH. Contiene 256 diferentes vectores de interrupción de 4 bytes. Un vector de interrupción contiene la dirección (segmento y desplazamiento) del procedimiento de servicio de la interrupción.

En la figura 10-2 se ilustra la tabla de vectores de interrupción para el microprocesador. Los cinco primeros vectores de interrupción son idénticos en todos los microprocesadores Intel, desde el 8086 hasta el 80486. Hay otros vectores de interrupción que tienen compatibilidad ascendente con el 80386 o el 80486, pero no tienen compatibilidad descendente para el 8086 o el 8088. Intel reserva los primeros 32 vectores de interrupción para emplearlos en sus diversos microprocesadores. Los últimos 224 vectores están disponibles como vectores de interrupción para el usuario. Cada vector tiene 4 bytes de longitud y contiene la dirección inicial del procedimiento de servicio de la interrupción. Los dos primeros bytes del vector contienen la dirección del desplazamiento, y los dos últimos la dirección del segmento.

En la siguiente lista se describe la función de cada interrupción dedicada del microprocesador.

1. Tipo 0. Error al dividir: ocurre siempre que hay sobreflujo en el resultado de una división o siempre que se intenta dividir entre cero.
2. Tipo 1. Modo paso a paso o de trampa: ocurre después de la ejecución de cada instrucción si está activo el bit (TF) de la bandera de trampa. Al aceptar esta interrupción, se borra el bit TF de modo que el procedimiento de servicio de la interrupción se ejecuta a toda velocidad. Más adelante en este capítulo se dan detalles adicionales.
3. Tipo 2. Interrupción de hardware no enmascarable: es el resultado de poner un 1 lógico en la terminal de entrada NMI del microprocesador. Esta entrada no se puede enmascarar, lo cual significa que no se puede deshabilitar.
4. Tipo 3. Interrupción de un byte: es la instrucción especial (INT 3) de un byte que emplea este vector para accesar el procedimiento de servicio de la interrupción. La instrucción INT 3 se emplea con frecuencia para almacenar un punto de ruptura en un programa para depurarlo.
5. Tipo 4: Sobreflujo: es un vector especial utilizado con la instrucción INTO, la cual interrumpe el programa si existe un sobreflujo señalado por la activación de esa bandera de desbordamiento (OF).
6. Tipo 5. BOUND: es una instrucción que compara un registro con límites almacenados en la memoria. Si el contenido del registro es igual que o mayor que la primera palabra en la memoria y menor o igual que la segunda palabra, no ocurre interrupción porque el contenido del registro está dentro de límites. Si el contenido del registro está fuera de esos límites, ocurre una interrupción de tipo 5.
7. Tipo 6. Código inválido: ocurre siempre que se encuentra un código de instrucción indefinido en un programa.

FIGURA 10–2 (a) La tabla de vectores de interrupción para el microprocesador. (b) El contenido de un vector de interrupción.



(a)

Cualquier vector de interrupción

3	Segmento (alto)
2	Segmento (bajo)
1	Desplazamiento (alto)
0	Desplazamiento (bajo)

(b)

8. Tipo 7. Coprocesador no disponible: ocurre cuando no se encuentra un coprocesador en el sistema, como lo ordenan los bits de control del coprocesador en la palabra (MSW) de estado de la máquina. Si se ejecuta una instrucción ESC o WAIT y no se encuentra el coprocesador, ocurre una interrupción o excepción de tipo 7.
9. Tipo 8. Doble falta: se activa siempre que ocurren dos interrupciones separadas durante la misma instrucción.
10. Tipo 9. Sobreflujo de segmento del coprocesador: ocurre si la instrucción ESC (código para el coprocesador) se extiende más allá de la dirección de desplazamiento FFFFH.
11. Tipo 10. Segmento de estado de tarea no válido: ocurre si el TSS es no válido debido a que el campo de límite del segmento no es de 002BH o mayor. En la mayor parte de los casos, se debe a que el TSS no está inicializado.
12. Tipo 11. Segmento no presente: ocurre cuando el bit P (P =0) en un descriptor indica que el segmento no está presente o que no es válido.
13. Tipo 12. Desborda del segmento de pila: ocurre si el segmento de pila no está presente (P = 0) o si se ha excedido el límite del segmento de pila.
14. Tipo 13. Protección general: ocurre para la mayor parte de las violaciones de la protección en el sistema de modo protegido de 80286, 80386 y 80486. A continuación aparece una lista de estas violaciones:
    a. Se ha excedido el límite de la tabla de descriptores
    b. Se han violado las reglas del privilegio
    c. Se cargó un tipo no válido para el segmento del descriptor
    d. Escribir en el segmento de código que está protegido
    e. Leer el segmento de sólo ejecutar código
    f. Escribir el segmento de sólo leer datos
    g. Se ha excedido del límite del segmento
    h. CPL ≠ 0, al ejecutar CTS, HLT, LGDT, LIDT, LLDT, LMSW o LTR
    i. CPL > IOPL al ejecutar CLI, IN, INS, LOCK, OUT, OUTS y STI
15. Tipo 14. Falla de acceso a página: ocurre cuando falla el acceso a una página de memoria en el 80386 y el 80486.
16. Tipo 16. Error del coprocesador: sucede cuando ocurre un error del coprocesador ($\overline{ERROR}$ = 0) para las instrucciones ESC y WAIT sólo en el microprocesador 80386.

Instrucciones para interrupción: BOUND, INTO, INT, INT 3 e IRET

Entre las cinco instrucciones de interrupción disponibles para el microprocesador, INT e INT 3 son muy semejantes. BOUND e INTO son condicionales e IRET es una instrucción especial para el retorno de interrupción.

La instrucción BOUND, que tiene dos operandos, compara un registro con dos palabras de datos de la memoria. Por ejemplo, si se ejecuta la instrucción BOUND AX,DATA, se compara a AX con el contenido de DATA y de DATA + 1 y también con DATA +2 y DATA + 3. Si AX es menor que el contenido de DATA y de DATA + 1 ocurre una interrupción de tipo 5. Si AX es mayor que DATA + 2 y DATA + 3, ocurre una interrupción del tipo 5. Si AX está dentro de los límites de estas dos palabras de la memoria, no ocurre interrupción.

La instrucción INTO comprueba la bandera de sobreflujo (OF). Si OF = 1, la instrucción INTO llama al procedimiento cuya dirección está almacenada en el vector de interrupción tipo 4.

Si OF = 0, entonces la instrucción INTO efectúa una NOP y la siguiente instrucción, en secuencia, del programa.

La instrucción INT n, llama al procedimiento de servicio de interrupción que comienza en la dirección representada en el vector número n. Por ejemplo, una INT 80H o una INT 128 llama al procedimiento de servicio de interrupción cuya dirección está almacenada en el vector 80H (000200H–000203H.) Para determinar la dirección del vector, se multiplica el número del vector (n) por 4. Con ello, se tiene la dirección inicial del vector de interrupción de cuatro bytes de longitud. Por ejemplo, una INT 5 = 4 × 5 o 20 (14H). El vector para INT 5 empieza en la dirección 000014H y continúa hasta 000017H. Cada instrucción INT se almacena en dos bytes de memoria en que el primer byte contiene el código y, el segundo, el número de la interrupción. La única excepción es la instrucción INT 3 que es de un byte y la cual se utiliza a menudo como interrupción de punto de ruptura, porque es fácil intercalar una instrucción de un byte en un programa. Los puntos de ruptura se utilizan a menudo para depurar programas.

La instrucción IRET es una instrucción especial para retorno y se utiliza para retornar de las interrupciones de software y hardware. La instrucción IRET es como una instrucción RET normal lejana, porque recupera la dirección de retorno de la pila. Es diferente porque también recupera una copia del registro de banderas de la pila. La instrucción IRET extrae 6 bytes de la pila, 2 para IP, 2 para CS y 2 para las banderas.

En los microprocesadores 80386 y 80486 hay también una instrucción IRETD porque éstos salvan el registro EFLAG (32 bits) en la pila, así como el EIP de 32 bits en el modo protegido. Cuando se trabaja en el modo real, se emplea la instrucción IRET con dichos microprocesadores.

El funcionamiento de una interrupción

Cuando el microprocesador concluye de ejecutar la instrucción en curso, entonces para determinar si una interrupción está activa comprueba: (1) alguna instrucción en ejecución; (2) trampa; (3) NMI; (4) sobreflujo del segmento del coprocesador; (5) INTR; (6) la instrucción INT en el orden citado. Si está presente una o más de estas condiciones de interrupción, ocurre lo siguiente:

1. Se salva el contenido del registro de banderas en la pila.
2. Se desactivan las banderas de interrupción (IF) y de trampa (TF). Esto deshabilita la terminal INTR y también la caracteristica de trampa.
3. Se salva el contenido del registro (CS) de segmento de código hacia la pila.
4. Se salva el contenido del apuntador de instrucción (IP) en la pila.
5. Se recupera el contenido del vector de la interrupción y se coloca en IP y en CS, de modo que la siguiente instrucción se ejecute en el procedimiento de servicio de la interrupción direccionado por el vector.

Siempre que se acepta una interrupción, el microprocesador salva en la pila el contenido del registro de banderas, de CS e IP, desactiva IF y TF y brinca al procedimiento direccionado por el vector de la interrupción. Después de salvar las banderas en la pila, se desactivan IF y TF. Estas banderas vuelven al estado que tenían antes de la interrupción, cuando la instrucción IRET se encuentra al final del procedimiento del servicio de la interrupción. Por tanto, si se habilitaron las interrupciones antes del procedimiento de servicio de la interrupción, la instrucción IRET al final del procedimiento, las vuelve a habilitar en forma automática.

Se salva la dirección de retorno (en IP y CS) en la pila durante la interrupción. A veces la dirección de retorno apunta hacia la siguiente instrucción en el programa y, a veces, apunta hacia la instrucción o punto en el programa en que ocurrió la interrupción. Las interrupciones números 0, 5, 6, 7, 8, 10, 11, 12 y 13 activa la dirección de retorno de la instrucción que ocasionó la interrupción, en vez de hacerlo a la siguiente instrucción en el programa. Esto permite la posibilidad de que el procedimiento de servicio de interrupción vuelva a intentar probar la instrucción en ciertos casos de error.

Algunas de las interrupciones en el modo protegido (tipos 8, 10, 11, 12 y 13) ponen un código de error en la pila después de la dirección de retorno. El código de error determina al selector que ocasionó la interrupción. En casos en que no interviene un selector, el código de error es 0.

Bits de bandera de interrupción

La bandera de interrupción (IF) y la bandera de trampa (TF) se borran una vez que se salvan en la pila el contenido del registro de banderas durante una interrupción. En la figura 10-3 se ilustra el contenido del registro de banderas y la posición de IF y TF. Cuando se activa el bit IF, permite que la terminal INTR ocasione una interrupción; cuando se desactiva el bit IF impide que la terminal INTR produzca una interrupción. Cuando TF = 1, ocasiona una interrupción de trampa (tipo número 1) cada vez de que se ejecuta una instrucción. Esa es la razón por la cual a menudo, se llama a la trampa paso a paso. Cuando TF = 0, ocurre la ejecución normal del programa. Este bit del registro de banderas permite la depuración en el 80386 y en el 80486 como se describe en el capítulo 14.

Las instrucciones STI y CLI activan y desactivan, respectivamente, interrupción. No hay instrucciones especiales para la bandera de trampa. En el ejemplo 10-1 se muestra un procedimiento de servicio de interrupción, que da comienzo al seguimiento al activar el bit de la bandera de trampa en la pila, dentro del procedimiento. En el ejemplo 10-2 se presenta un procedimiento de servicio de interrupción que desactiva el trazado porque borra la bandera de trampa en la pila dentro del procedimiento.

EJEMPLO 10–1

```
                          ;Procedimiento que activa a TF para habilitar la trampa
                          ;
0000                      TRON        PROC   FAR

0000 50                               PUSH AX                  ;salvar registros
0001 55                               PUSH BP

0002 8B EC                            MOV  BP,SP               ;obtener SP
0004 8B 46 08                         MOV  AX,[BP+8]           ;obtener banderas
0007 80 CC 01                         OR   AH,1                ;activa TF
000A 89 46 08                         MOV  [BP+8],AX           ;salvar banderas
```

FIGURA 10–3 El registro de banderas del 80286. (Cortesía de Intel Corporation.)

BANDERAS																
	15–12	O	D	I	T	S	Z		A		P		C
	15	11	10	9	8	7	6	5	4	3	2	1	0

```
                                                    ;restaurar registros
000D 5D                       POP  BP
000E 58                       POP  AX
000F CF                       IRET

0010          TRON            ENDP
```

EJEMPLO 10–2

```
              ;Procedimiento que desactiva TF para deshabilitar la trampa
              ;
0010          TROFF           PROC    FAR
                                                    ;salvar registros
0010 50                       PUSH AX
0011 55                       PUSH BP

0012 8B EC                    MOV  BP,SP            ;obtener SP
0014 8B 46 08                 MOV  AX,[BP+8]        ;obtener banderas
0017 80 E4 FE                 AND  AH,0FEH          ;borrar TF
001A 89 46 08                 MOV  [BP+8],AX        ;salvar banderas
                                                    ;restaurar registros
001D 5D                       POP  BP
001E 58                       POP  AX
001F CF                       IRET

0020          TROFF           ENDP
```

En ambos ejemplos, se recupera el registro de banderas de la pila con el empleo del registro BP, que en forma implícita direcciona al segmento de pila. Una vez recuperadas las banderas, se establece (TREN) el bit TF o se lo borra (TROAPG) antes de retornar el procedimiento de servicio de interrupción. La instrucción IRET recupera el registro de bandera con el nuevo estado de la bandera de trampa.

Procedimiento para trazar. Se supone que una instrucción INT 40H accesa a TREN y que una instrucción 41H acede a TRAPG. En el ejemplo 10-3 retraza un programa que sigue de la instrucción 40H. El procedimiento de servicio de la interrupción ilustrado en el ejemplo 10-3, responde a la interrupción tipo 1 o a una interrupción de trampa. Cada vez que ocurre así, después de que se ejecuta cada instrucción que sigue de INT 40H, el procedimiento TRAZA exhibe el contenido de todos los registros de 16 bits del microprocesador en el monitor. Con ello, se "traza" el contenido de los registros para todas las instrucciones entre la INT 40H (TREN) y la INT 41H (TRAPG).

EJEMPLO 10–3 (página 1 de 3)

```
              ;Procedimiento que exhibe todos los registros y su contenido
              ;en la pantalla del monitor, en respuesta a una interrupción de trampa
              ;
0000          TRAZA   PROC    FAR

0000 50               PUSH AX                       ;salvar registros
0001 55               PUSH BP
0002 53               PUSH BX

0003 BB 0054 R        MOV  BX,OFFSET NAMES          ;direccionar nombres de registros

              ;exhibir registros
```

EJEMPLO 10-3 (página 2 de 3)

```
0006 E8 009A R              CALL CRLF          ;obtener nueva línea de exhibición
0009 E8 00A9 R              CALL DISP          ;exhibir AX
000C 8B C3                  MOV  AX,BX
000E E8 00A9 R              CALL DISP          ;exhibir BX
0011 8B C1                  MOV  AX,CX
0013 E8 00A9 R              CALL DISP          ;exhibir CX
0016 8B C2                  MOV  AX,DX
0018 E8 00A9 R              CALL DISP          ;exhibir DX
001B 8B C4                  MOV  AX,SP
001D 05 000C                ADD  AX,12
0020 E8 00A9 R              CALL DISP          ;exhibir SP
0023 8B C5                  MOV  AX,BP
0025 E8 00A9 R              CALL DISP          ;exhibir BP
0028 8B C6                  MOV  AX,SI
002A E8 00A9 R              CALL DISP          ;exhibir SI
002D 8B EC                  MOV  BP,SP         ;direccionar pila con BP
002F 8B 46 06               MOV  AX,[BP+6]
0032 E8 00A9 R              CALL DISP          ;exhibir IP
0035 8B 46 0A               MOV  AX,[BP+10]
0038 E8 00A9 R              CALL DISP          ;exhibir banderas
003B 8B 46 08               MOV  AX,[BP+8]
003E E8 00A9 R              CALL DISP          ;exhibir CS
0041 8C D8                  MOV  AX,DS
0043 E8 00A9 R              CALL DISP          ;exhibir DS
0046 8C C0                  MOV  AX,ES
0048 E8 00A9 R              CALL DISP          ;exhibir ES
004B 8C D0                  MOV  AX,SS
004D E8 00A9 R              CALL DISP          ;exhibir SS
0050 5B                     POP  BX            ;restaurar registros
0051 5D                     POP  BP
0052 58                     POP  AX
0053 CF                     IRET

0054                TRACE   ENDP

0054 41 58 20 3D 20 NAMES   DB   'AX = '
0059 42 58 20 3D 20         DB   'BX = '
005E 43 58 20 3D 20         DB   'CX = '
0063 44 58 20 3D 20         DB   'DX = '
0068 53 50 20 3D 20         DB   'SP = '
006D 42 50 20 3D 20         DB   'BP = '
0072 53 49 20 3D 20         DB   'SI = '
0077 44 49 20 3D 20         DB   'DI = '
007C 49 50 20 3D 20         DB   'IP = '
0081 46 4C 20 3D 20         DB   'FL = '
0086 43 53 20 3D 20         DB   'CS = '
008B 44 53 20 3D 20         DB   'DS = '
0090 45 53 20 3D 20         DB   'ES = '
0095 53 53 20 3D 20         DB   'SS = '

009A                CRLF    PROC      NEAR

009A 50                     PUSH AX            ;salvar registros
009B 52                     PUSH DX

009C B4 06                  MOV  AH,6
009E B2 0D                  MOV  DL,13
00A0 CD 21                  INT  21H           ;exhibir CR
00A2 B2 0A                  MOV  DL,10
00A4 CD 21                  INT  21H           ;exhibir LF
```

EJEMPLO 10–3 (página 3 de 3)

```
00A6 5A                    POP  DX              ;restaurar registros
00A7 58                    POP  AX
00A8 C3                    RET

00A9             CRLF      ENDP

00A9             DISP      PROC      NEAR

00A9 52                    PUSH DX              ;salvar registros
00AA 57                    PUSH DI
00AB 51                    PUSH CX
00AC 50                    PUSH AX

00AD B4 06                 MOV  AH,6
00AF B9 0005               MOV  CX,5

00B2             DISP1:

00B2 2E: 8A 17             MOV  DL,CS:[BX]      ;exhibir el nombre
00B5 CD 21                 INT  21H
00B7 43                    INC  BX
00B8 E2 F8                 LOOP DISP1
OOBA 5F                    POP  DI              ;obtener valor numérico
00BB 57                    PUSH DI
00BC B6 04                 MOV  DH,4            ;establecer conteo

00BE             DISP2:

00BE B9 0004               MOV  CX,4
00C1 D3 C7                 ROL  DI,CL
00C3 8B C7                 MOV  AX,DI
00C5 B4 06                 MOV  AH,6
00C7 8A D0                 MOV  DL,AL
00C9 80 E2 0F              AND  DL,15
00CC 80 C2 30              ADD  DL,30H          ;convertir a ASCII
00CF 80 FA 39              CMP  DL,39H
00D2 76 03                 JBE  DISP3
00D4 80 C2 07              ADD  DL,7

00D7             DISP3:

00D7 CD 21                 INT  21H
00D9 FE CE                 DEC  DH
00DB 75 E1                 JNZ  DISP2           ;repetir para 4 dígitos

00DD B4 06                 MOV  AH,6
00DF B2 20                 MOV  DL,' '          ;exhibir espacio
00E1 CD 21                 INT  21H

00E3 58                    POP  AX              ;restaurar registros
00E4 59                    POP  CX
00E5 5F                    POP  DI
00E6 5A                    POP  DX
00E7 C3                    RET

00E8             DISP      ENDP
```

Almacenamiento de un vector de interrupción en la tabla de vectores

A fin de instalar un vector de interrupción —llamado a veces cambio de servicio—, el ensamblador debe direccionar a la memoria absoluta. En el ejemplo 10-4 se muestra la forma en que se agrega un nuevo vector a la tabla de vectores de interrupción con el empleo del ensamblador y una llamada a una función del DOS. En este caso, se llama a la función 25H de la INT 21H que inicializa el vector de interrupción. Se debe tener en cuenta que lo primero que se hace en este procedimiento es salvar el vector de interrupción anterior con la función 35H de la INT 21H del DOS, para leer el vector actual. En el apéndice A aparecen mayores detalles de las llamadas a funciones del DOS.

EJEMPLO 10–4

```
                              ;Procedimiento que instala un nuevo vector de interrupción
                              ;en la interrupción tipo número 40H
                              ;
0000 0000                     OLDOFF  DW    ?
0002 0000                     OLDSEG  DW    ?
= 0004                        NEWOFF  EQU   THIS WORD
= 0006                        NEWSEG  EQU   THIS WORD+2
0004 0200 ---- R              NEW     DD    FAR PTR TRON

0008                          IN_40H  PROC  NEAR

0008 06                               PUSH ES              ;salvar registros
0009 1E                               PUSH DS
000A 50                               PUSH AX
000B 53                               PUSH BX
000C 52                               PUSH DX

000D B4 35                            MOV  AH,35H          ;obtener vector actual
000F B0 40                            MOV  AL,40H          ;número y tipo
0011 CD 21                            INT  21H

0013 2E: 89 1E 0000 R                 MOV  CS:OLDOFF,BX    ;salvar desplazamiento anterior
0018 8C C0                            MOV  AX,ES
001A 2E: A3 0002 R                    MOV  CS:OLDSEG,AX    ;salvar segmento anterior

001E 2E: 8B 16 0004 R                 MOV  DX,NEWOFF       ;obtener desplazamiento nuevo
0023 2E: A1 0006 R                    MOV  AX,NEWSEG       ;obtener nuevo segmento
0027 8E D8                            MOV  DS,AX

0029 B4 25                            MOV  AH,25H          ;instalar nuevo vector
002B B0 40                            MOV  AL,40H          ;número y tipo
002D CD 21                            INT  21H

002F 5A                               POP  DX              ;restaurar registros
0030 5B                               POP  BX
0031 58                               POP  AX
0032 1F                               POP  DS
0033 07                               POP  ES
0034 C3                               RET

0035                          MIN_40H ENDP
```

10–2 INTERRUPCIONES DE PERIFERICOS

El microprocesador tiene dos terminales de entrada para interrupciones de periféricos: (hardware) de la interrupción no enmascarable (NMMI) y la de solicitud de interrupción (INTR). Siempre que se activa la entrada NMI ocurre una interrupción tipo 2 porque NMI se decodifica en el interior. La entrada INTR se debe decodificar externamente para seleccionar un vector. Se puede escoger cualquier vector de interrupción para la terminal INTR, pero por lo general se utiliza una interrupción entre 20H y FFH. La señal $\overline{\text{INTA}}$ es también una terminal relacionada con las interrupciones en el microprocesador, pero es una salida que se utiliza en respuesta a la entrada INTR para aplicar un número de vector en las terminales D7–D0 del canal de datos. En la figura 10-4 se ilustran las tres conexiones de interrupción del microprocesador.

NMI

La interrupción no enmascarable (NMI) es una entrada disparada por flanco que solicita una interrupción en el flanco positivo (transición de 0 a 1). Después de un borde positivo, la terminal NMI debe permanecer como 1 lógico hasta que la reconozca el microprocesador. Se debe tener en cuenta que antes de que se reconozca el flanco positivo, la terminal NMI debe ser un 0 lógico durante, cuando menos, dos periodos de reloj.

La entrada NMI se utiliza a menudo para errores de paridad y otros problemas graves en el sistema, como las interrupciones de energía. Las interrupciones en la energía eléctrica se pueden detectar con facilidad si se vigila la línea de alimentación y se ocasiona una interrupción de NMI cada vez que se interrumpe la energía. En respuesta a este tipo de interrupción, el microprocesador almacena todos los registros internos en una memoria con respaldo de pilas o sea una EEPROM. En la figura 10-5 se ilustra un circuito de detección de interrupción de energía, que envía un 1 lógico a la entrada NMI cada vez que se interrumpe la corriente.

En este circuito, un aislador óptico aísla de la línea de CA. La salida del aislador se forma con un inversor Schmitt, que produce un pulso de 60 Hz a la entrada de disparo del multivibrador monoestable 74LS122, que se puede volver a disparar. Los valores de R y C se escogen de modo que el 74LS122 produzca un pulso activo de 33 ms en 2 periodos de CA. Debido a que el 74LS122

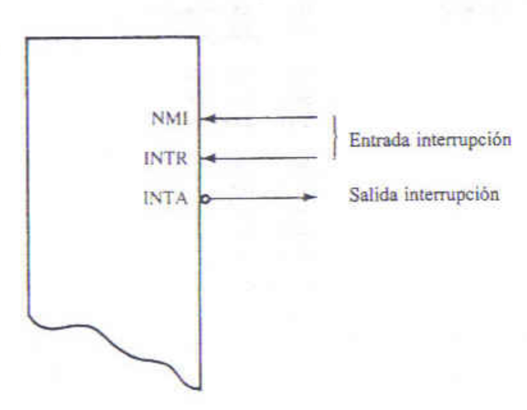


FIGURA 10–4 Terminales de interrupción del 80286.

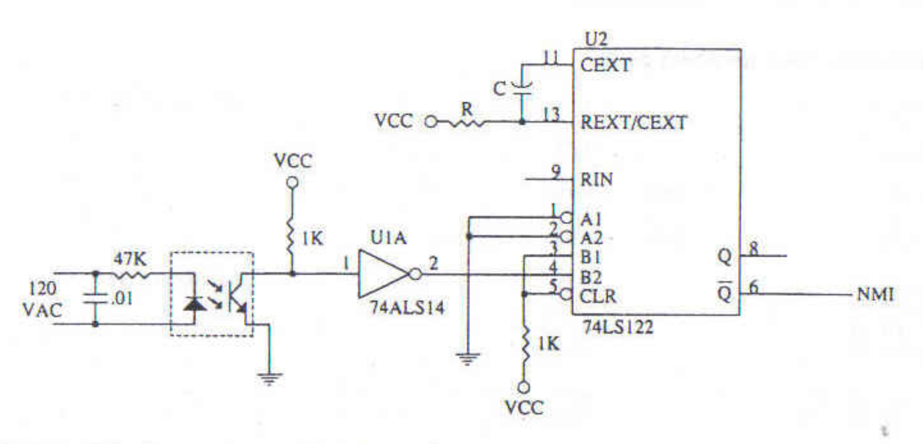


FIGURA 10–5 Circuito para detectar falla de la energía.

se puede volver a disparar, mientras haya CA, la salida $\overline{Q}$ permanece disparada como 1 lógico y Q permanece como 0 lógico.

Si se interrumpe la corriente, el 74LS122 ya no recibe pulsos de disparo desde el 74ALS14, lo cual significa que Q retorna a 0 lógico y $\overline{Q}$ retorna a 1 lógico e interrumpe al microprocesador por medio de la terminal NMI. El procedimiento de servicio de interrupción, que no se ilustra, almacena el contenido de todos los registros internos y otros datos en una memoria respaldada con pilas. El sistema supone que la fuente de alimentación del sistema tiene un capacitor de filtro de suficiente tamaño para suministrar corriente cuando menos 75 ms después de que se interrumpe la corriente.

En la figura 10-6 se ilustra el circuito que alimenta a la memoria, cuando se interrumpe el suministro de corriente directa. Se utilizan diodos para conmutar los voltajes de alimentación del

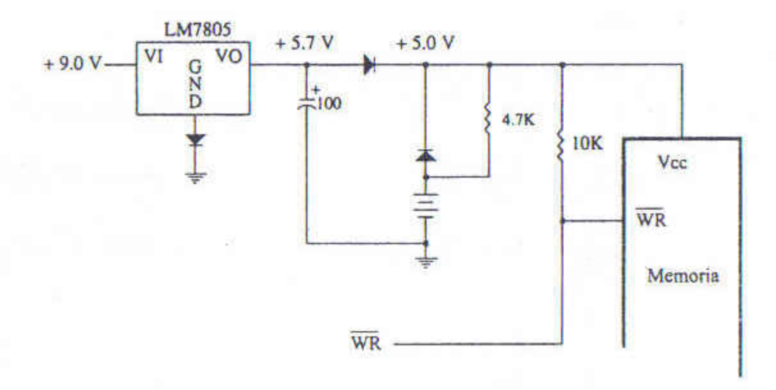


FIGURA 10–6 Sistema de memoria con respaldo de pilas con el empleo de pilas de NiCad, litio o celda de gel.

suministro de CD a la pila. Los diodos son normales, de silicio, porque el suministro de corriente a esta memoria es alto, de +5.0V a +5.7V. Se verá también que se emplea el resistor para la carga lenta de la pila, que puede ser Nicad, de litio o una celda de gel.

Cuando se interrumpe el suministro de CD, la pila aplica un voltaje reducido en la terminal Vcc de la memoria. Casi todas las memorias retendrán datos con voltajes Vcc de apenas 1.5V, por lo cual el voltaje de la pila no necesita ser de +5.0V. A la terminal $\overline{WR}$ se lleva a Vcc durante una interrupción de energía, para que no se escriban datos en la memoria.

INTR e INTA

La entrada de solicitud de interrupción (INTR) es sensible al nivel, lo cual significa que se la debe mantener a un valor de 1 lógico hasta que se la reconozca. Un acontecimiento externo activa la terminal INTR y se desactiva dentro del procedimiento de servicio de la interrupción. Esta entrada se deshabilita en forma automática una vez que la ha aceptado el microprocesador, y se vuelve a habilitar con la instrucción IRET al final del procedimiento de servicio de interrupción. En el 80386 y el 80486 se emplea la instrucción IRETD en el funcionamiento en modo protegido.

El microprocesador responde a la entrada INTR pulsando dos veces la salida $\overline{INTA}$ para recibir el número de vector de la interrupción en la conexión D7–D0 en el canal de datos. En la figura 10-7 se ilustra el diagrama de temporización para las terminales INTR e $\overline{INTA}$ del microprocesador. El sistema genera dos pulsos $\overline{INTA}$ que se emplean para insertar el número (tipo) de vector en el canal de datos.

En la figura 10-8 se ilustra un circuito sencillo que aplica el vector de interrupción FFH en el canal de datos en respuesta a una INTR. Se debe tener en cuenta que la terminal $\overline{INTA}$ no está conectada en este circuito. Debido a que se utilizan resistores para poner en alto las conexiones D0–D7 del canal de datos, entonces el microprocesador detecta en forma automática el número de vector FFH, en respuesta a INTR. Se debe tener en cuenta que la terminal INTA no está conectada en el circuito. Debido a que se utilizan resistores para poner en alto las conexiones

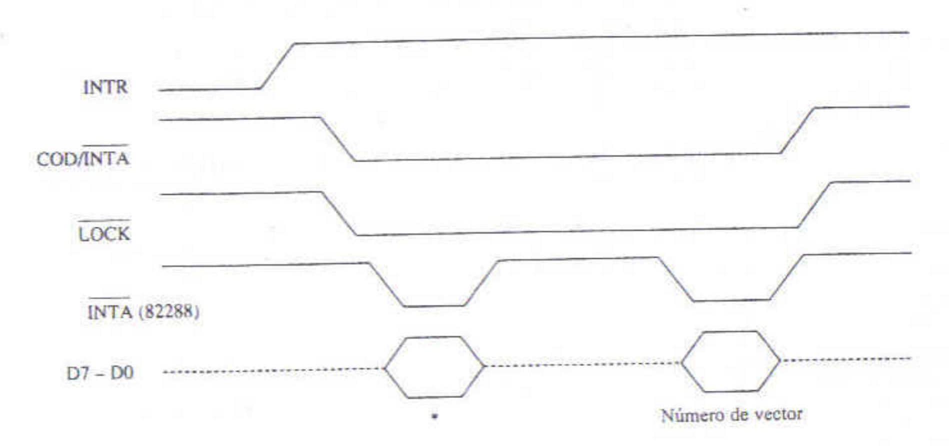


FIGURA 10–7 La temporización de los pulsos INTR y $\overline{INTA}$ del controlador de canal 82288. *NOTA: Esta parte del canal de datos no se tiene en cuenta y suele contener el número de vector.

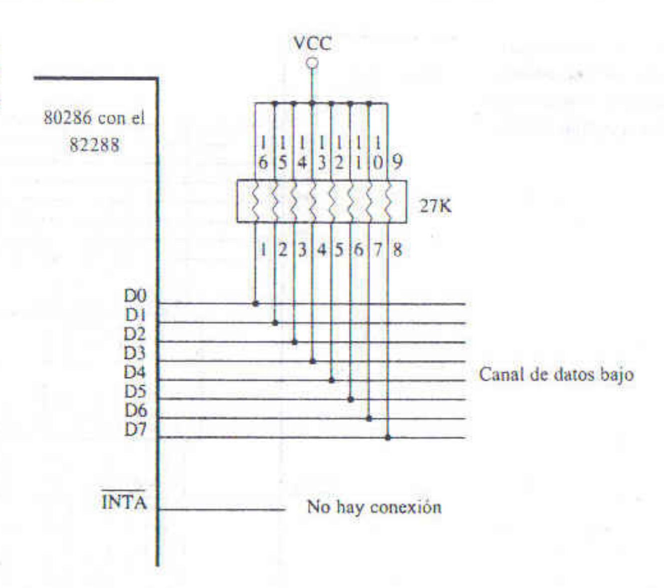


FIGURA 10–8 Un método sencillo para generar el vector número FFH de interrupción en respuesta a INTR.

D0–D7 del canal de datos, el microprocesador detecta en forma automática el número de vector FFH en respuesta a la entrada INTR. Esta es, quizá, la forma más económica de implantar la terminal INTR en el microprocesador.

Empleo de un acoplador de tres estados para $\overline{INTA}$. En la figura 10-9 se ilustra la forma en que el vector de interrupción 80H se coloca en el canal de datos D0–D7 en respuesta a INTR, la salida del microprocesador a $\overline{INTA}$ se emplea para habilitar un acoplador octal 74ALS244 de tres estados. Este aplica el número de vector de interrupción en el canal de datos en respuesta al pulso $\overline{INTA}$. El número de vector se cambia con facilidad con los interruptores DIP que se muestran en la ilustración.

Haciendo a la entrada INTR disponible por flanco. A menudo se necesita una entrada disparada por flanco en lugar de una entrada sensible al nivel La entrada INTR se puede convertir en entrada disparada por flanco con el empleo de un flip-flop tipo D, como se ilustra en la figura 10-10. En este caso, la entrada de reloj se convierte en una entrada de solicitud de interrupción disparada por flanco y la entrada de borrar se emplea para desactivar la solicitud cuando el microprocesador da salida la señal $\overline{INTA}$. También se debe tener en cuenta que la señal de reinicialización (RESET), al principio, borra el flip-flop para que no se solicite interrupción al aplicar energía al sistema por primera vez.

Interrupción de teclado con un 8255

El ejemplo del teclado del capítulo 9 es un sencillo ejemplo del funcionamiento de la entrada INTR y de una interrupción. En la figura 10-11 se ilustra la conexión del 8255 con el microproce-

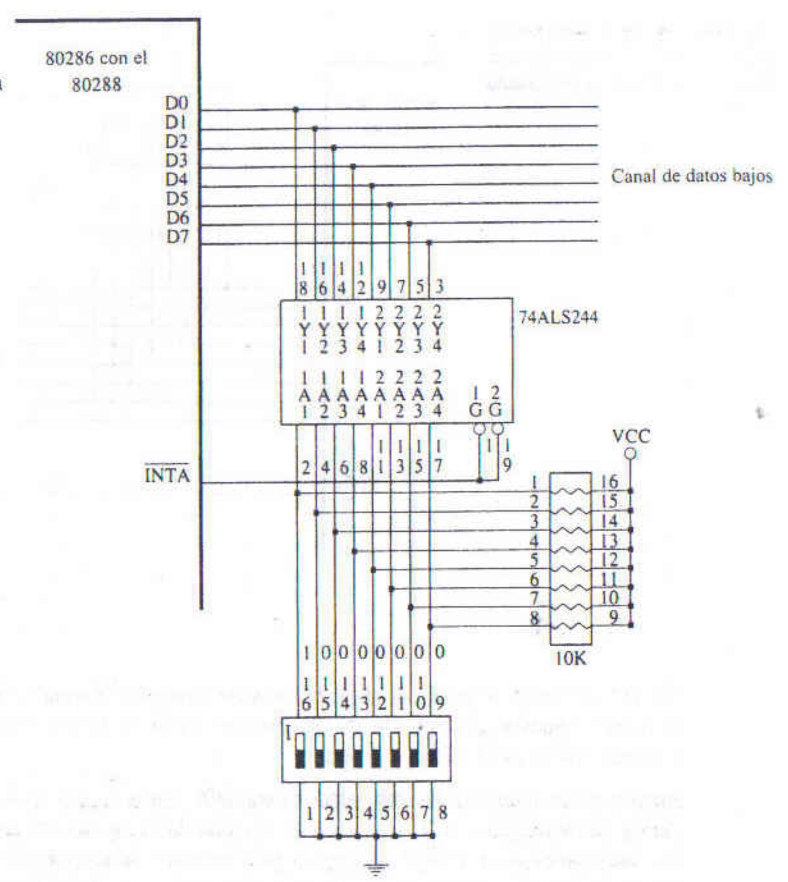


FIGURA 10–9 Un circuito que aplica cualquier número de vector de interrupción en respuesta a $\overline{\text{INTA}}$. Se ilustra la aplicación del 80H.

sador y el teclado. También se ilustra la forma en que se emplea un acoplador octal 74ALS244 para enviar al microprocesador el vector de interrupción 40H, en respuesta a una interrupción en el teclado durante el pulso $\overline{\text{INTA}}$.

El 8255 se decodifica en las direcciones 0500H, 0502H, 0504H y 0506H de puertos de E/S del 8086 con un PAL 16L8 (no se ilustra el programa). El 8255 funciona en el modo 1 (habilitación por entrada estroboscópica), por lo cual, cuando se oprime una tecla la entrada INTR (PC3) se vuelve un 1 lógico que solicita una interrupción en la terminal INTR en el microprocesador. La terminal INTR permanece en nivel alto hasta que se leen los datos ASCII en el puerto A. Es decir, cada vez que se oprime una tecla, el 8255 solicita una interrupción tipo 40H por medio de la terminal INTR. La señal $\overline{\text{DAV}}$ del teclado hace que se capturen los datos en el puerto A y también hace que INTR vaya a un 1 lógico.

En el ejemplo 10-5 se ilustra el procedimiento del servicio de la interrupción para el teclado. Es muy importante salvar todos los registros que va a utilizar el servicio de la interrupción, antes

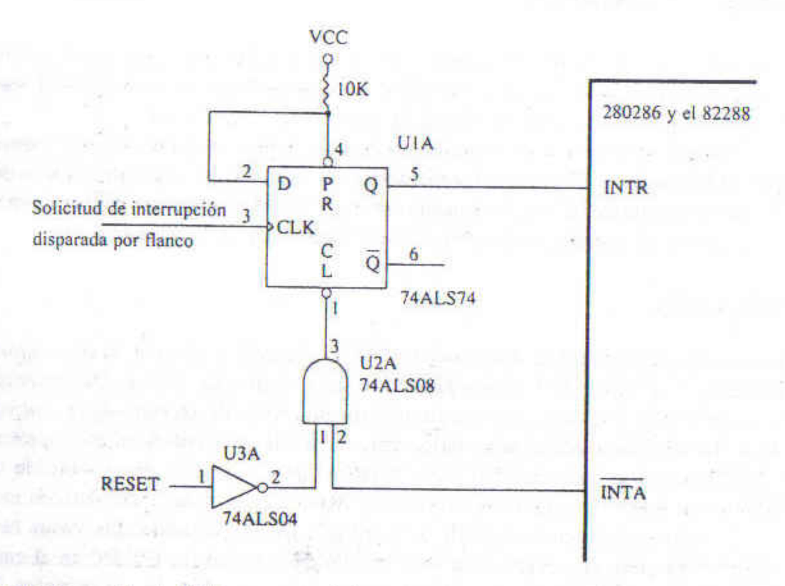


FIGURA 10–10 Conversión de INTR a una entrada de solicitud de interrupción disparada por flanco.

de usarlos. En el programa de inicialización del 8255 (que no se ilustra), se inicializa la FIFO de modo que ambos apuntadores sean iguales, se habilita la terminal INTR con el bit INTE dentro del 8255 y se programa el modo de funcionamiento.

EJEMPLO 10–5 (página 1 de 3)

```
                        ;Procedimiento de servicio de interrupción que lee
                        ;una tecla del teclado
= 0500                  PORTA   EQU   500H              ;puerto A
= 0506                  CNTR    EQU   506H              ;registro de control
0000 0100[              FIFO    DB    256 DUP (?)       ;memoria FIFO
          ??
                 ]

0100 0000               INP     DW    ?                 ;apuntador de entrada
0102 0000               OUTP    DW    ?                 ;apuntador de salida

0104                    TECLA   PROC   FAR
0104 50                         PUSH AX                 ;salvar registros
0105 53                         PUSH BX
0106 57                         PUSH DI
0107 52                         PUSH DX

0108 2E: 8B 1E 0100 R           MOV   BX,INP            ;direccionar FIFO
010D 2E: 8B 3E 0102 R           MOV   DI,OUTP
0112 FE C3                      INC   BL                ;probar si está llena,
0114 3B DF                      CMP   BX,DI
0116 74 11                      JE    FULL              ;si es que está llena
```

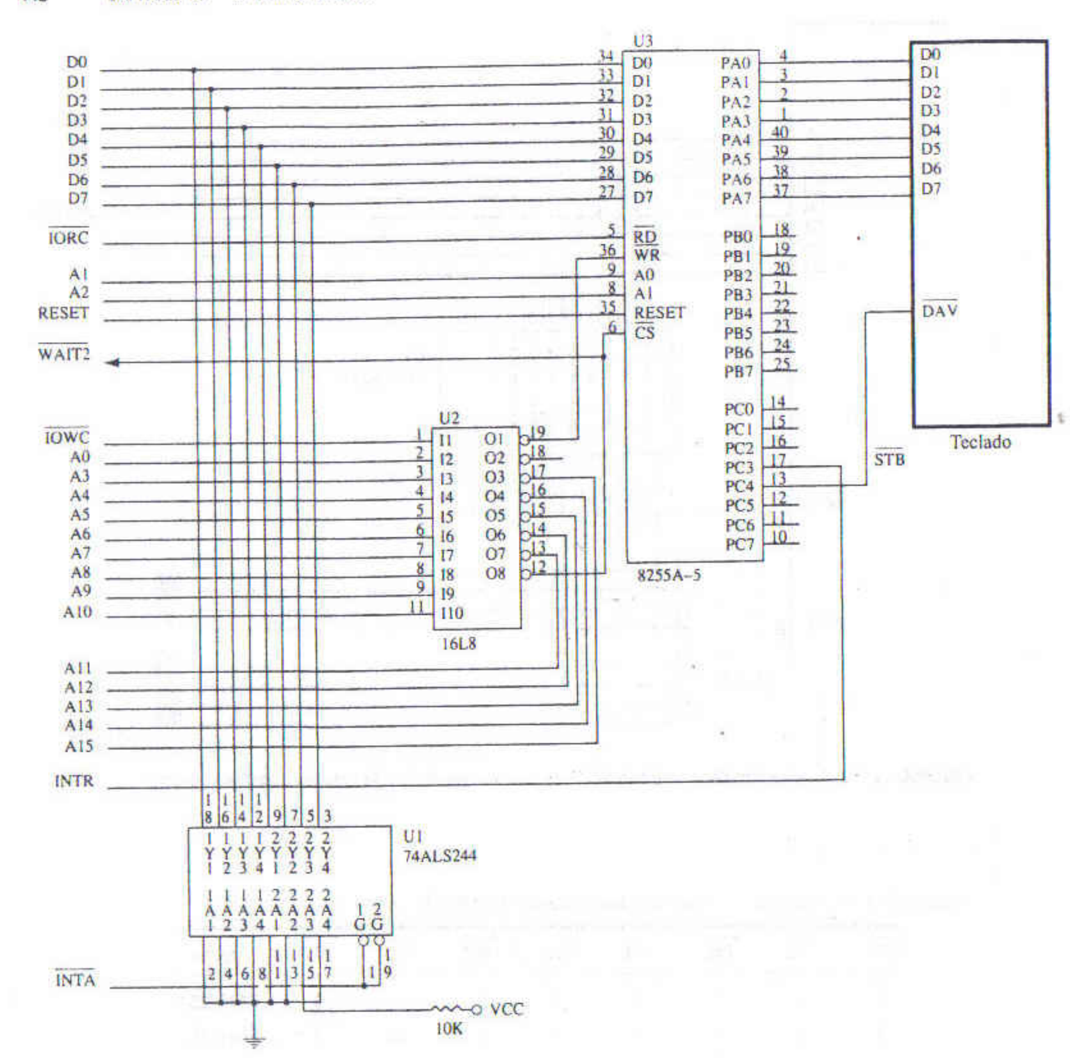


FIGURA 10–11 Una 8255A–5 en interface con un teclado en el sistema del 80286 con el empleo del vector de interrupción 40H.

EJEMPLO 10–5 (página 2 de 3)

```
0118 FE CB                DEC  BL
011A BA 0500              MOV  DX,PORTA
011D EC                   IN   AL,DX              ;obtener los datos
011E 2E: 88 07            MOV  CS:[BX],AL         ;salvar los datos
0121 2E: FE 06 0100 R     INC  BYTE PTR INP
0126 EB 07 90             JMP  DONE
```

EJEMPLO 10–5 (página 3 de 3)

```
0129                LLENA:

0129 B0 08                  MOV  AL,8             ;deshabilitar las interrupciones
012B BA 0506                MOV  DX,CNTR
012E EE                     OUT  DX,AL

012F                HECHO:

012F 5A                     POP  DX               ;restaurar registros
0130 5F                     POP  DI
0131 5B                     POP  BX
0132 58                     POP  AX
0133 CF                     IRET

0134                TECLA   ENDP
```

El procedimiento es bastante corto porque el 8086 ya "sabe" que los datos del teclado están disponibles cuando se llama al procedimiento. Se da entrada a los datos del teclado y se almacenan en una región de memoria tipo FIFO (primero en entrar, primero en salir). La mayor parte de las interfaces para teclado incluyen una FIFO de, cuando menos, 16 bytes. La FIFO de este ejemplo es de 256 bytes, lo cual es más que adecuado para una interface de teclado.

El procedimiento comprueba primero si la FIFO está llena. Se indica que está llena cuando el punto de entrada (ENT) está un byte debajo del apuntador de salida (SALI). Si la FIFO está llena, se deshabilita la interrupción con un comando de habilitar/deshabilitar bit al 8255 y ocurre un retorno de la interrupción. Si la FIFO no está llena, se da entrada a los datos del puerto A y se incrementa el apuntador de entrada antes de que ocurra un retorno.

En el ejemplo 10-6 se presenta el procedimiento para leer los datos de la FIFO. Este procedimiento determina primero si la FIFO está vacía, mediante la comparación de los dos apuntadores. Si los apuntadores son iguales, la FIFO está vacía y el programa espera en el ciclo vacío en donde prueba los apuntadores en forma continua. El ciclo vacío se interrumpe con la interrupción del teclado, que almacena datos en la FIFO para que ya no esté vacía. Este procedimiento retorna con el caracter en el registro AH.

EJEMPLO 10–6

```
                    ;Procedimiento que lee una tecla en la FIFO
                    ;y regresa con ella en AH
                    ;
0104                LEE     PROC   FAR

0104 53                     PUSH BX               ;salvar registros
0105 57                     PUSH DI
0106 52                     PUSH DX

0107                VACIO:

0107 2E: 8B 1E 0100 R       MOV  BX,INP           ;direccionar FIFO
010C 2E: 8B 3E 0102 R       MOV  DI,OUTP
0111 3B DF                  CMP  BX,DI            ;probar si está vacía
0113 74 F2                  JE   EMPTY            ;si es que está vacía
```

```
0115 2E: 8A 25              MOV  AH,CS:[DI]       ;obtener datos

0118 B0 09                  MOV  AL,9             ;habilitar interrupcion del 8255A
011A BA 0506                MOV  DX,CNTR
011D EE                     OUT  DX,AL

011E 2E: FE 06 0102 R       INC  BYTE PTR OUTP    ;incrementar apuntador

0123 5A                     POP  DX               ;restaurar registros
0124 5F                     POP  DI
0125 5B                     POP  BX
0126 CB                     RET

0127                  LEE   ENDP
```

10–3 EXPANSION DE LA ESTRUCTURA DE INTERRUPCION

En esta sección se describen tres de los métodos más comunes para ampliar o expandir la estructura de interrupciones del microprocesador. En esta sección se explica la forma en que, con programación y ciertas modificaciones en el circuito de la figura 10-9, es posible ampliar la entrada INTR de modo que acepte 7 entradas de interrupción. También se explica la forma de interrupción por cadena de margaritas mediante un programa de "encuesta". En la siguiente sección se describe una tercera técnica con la cual se pueden agregar hasta 63 entradas de interrupción por medio del controlador programable de interrupciones 8259A.

Empleo del 74ALS244 para ampliar interrupciones

La modificación ilustrada en la figura 10-12, permite que el circuito de la figura 10-9 acepte hasta siete entradas adicionales de interrupción. El único cambio en el circuito es la adición de una compuerta NAND con 8 entradas, que produce la señal INTR para el microprocesador cuando se activa cualquiera de las entradas $\overline{IR}$.

Funcionamiento. Si cualquiera de las entradas $\overline{IR}$ se convierte en 0 lógico, entonces la salida de la puerta NAND será un 1 lógico y solicitará una interrupción por medio de la entrada INTR. Cuál es el vector de interrupción que se busque y se cargue durante el pulso $\overline{INTA}$, depende de la línea que solicitó la interrupción. En la tabla 10-1 se muestran los vectores de interrupción utilizados por una sola entrada de solicitud de interrupción.

Si se activan al mismo tiempo dos o más entradas de solicitud de interrupción, se genera un nuevo vector de interrupción. Por ejemplo, si $\overline{IR1}$ y $\overline{IR0}$ se activan al mismo tiempo, el vector de interrupción que se genera es FCH (252). En esta localidad se resuelve la prioridad. Si la entrada $\overline{IR0}$ va a tener la máxima prioridad, la dirección del vector para ella se almacena en la localidad del vector FCH. Se debe emplear toda la mitad superior de la tabla de vectores con sus 128 vectores de interrupción, a fin de poder aceptar todas las posibles condiciones de estas siete entradas de solicitud de interrupción. Esto parece ser un desperdicio, pero en muchas aplicaciones dedicadas, es un método que resulta económico para la ampliación de las interrupciones.

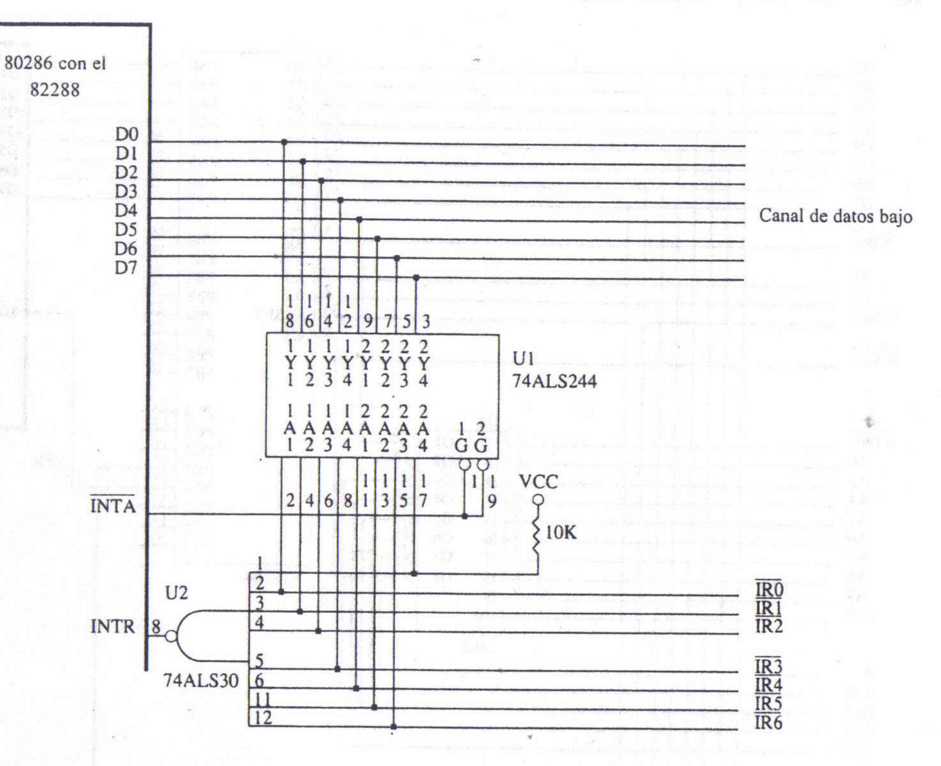


FIGURA 10–12 Ampliación de la entrada INTR de una a siete líneas de solicitud de interrupción.

TABLA 10–1 Solicitud sencilla de interrupción para la FIGURA 10–12

$\overline{AD7}$	$\overline{IR6}$	$\overline{IR5}$	$\overline{IR4}$	$\overline{IR3}$	$\overline{IR2}$	$\overline{IR1}$	$\overline{IR0}$	*Vector*
1	1	1	1	1	1	1	0	FEH (254)
1	1	1	1	1	1	0	1	FDH (253)
1	1	1	1	1	0	1	1	FBH (251)
1	1	1	1	0	1	1	1	F7H (247)
1	1	1	0	1	1	1	1	EFH (239)
1	1	0	1	1	1	1	1	DFH (223)
1	0	1	1	1	1	1	1	BFH (191)

Interrupciones en cadena de margarita

La expansión por medio de interrupciones en cadena de margaritas, es, en muchos aspectos, mejor que emplear la ampliación de interrupciones con un 74ALS244, porque sólo requiere un vector de interrupción. La tarea de determinar la prioridad se encomienda al procedimiento de servicio de la interrupción. Determinar la prioridad para una cadena de margaritas, requiere tiempo adicional de ejecución del programa, pero en general es un método mucho mejor que ampliar la estructura de interrupciones del microprocesador.

En la figura 10-13 se ilustra un conjunto de dos interfaces periféricas 8255, con sus cuatro entradas INTR en cadena de margaritas y conectadas a la entrada INTR del microprocesador. Si

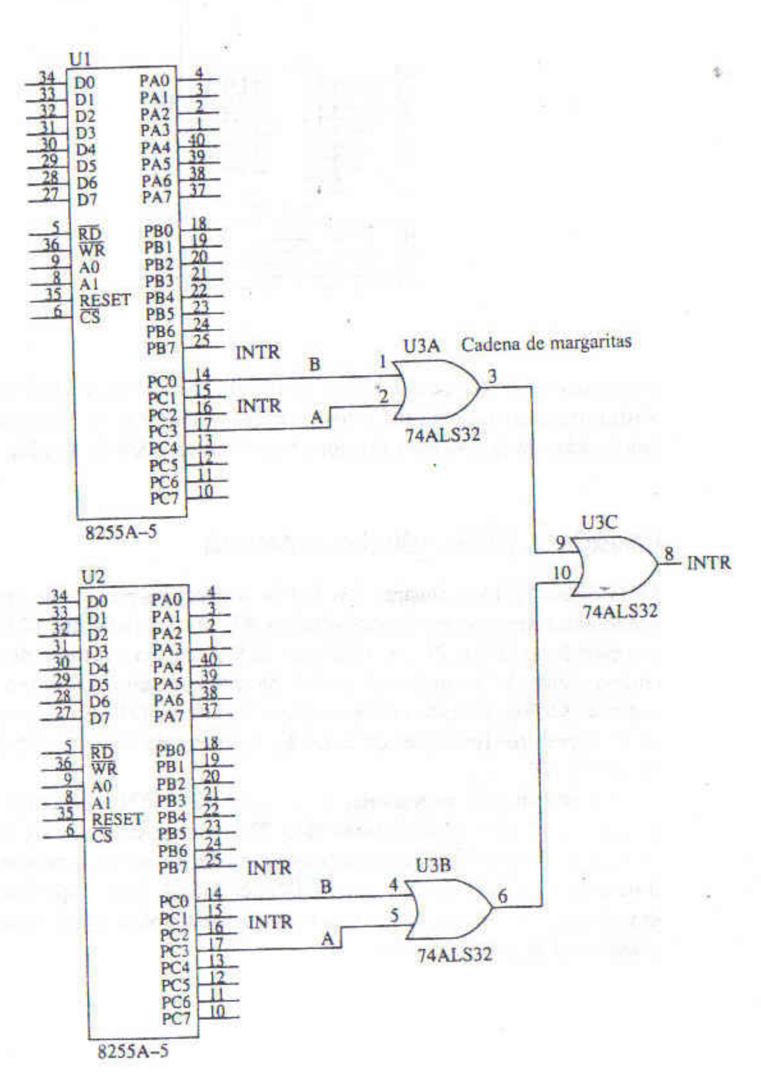


FIGURA 10–13 Dos PIA 8255A–5, conectados con las salidas INTR que están en cadena de margaritas para producir una señal INTR para el 80286.

cualquier interrupción se convierte en un 1 lógico, lo mismo hace la entrada INTR del microprocesador y ocasiona una interrupción.

Cuando se utiliza una cadena de margaritas para solicitar una interrupción, es mejor llevar a un nivel alto las conexiones D0–D7 del canal de datos, a fin de poder emplear el vector de interrupciones FFH para la cadena. En realidad, se puede emplear cualquier vector de interrupción para responder a la cadena de margaritas. En este circuito, cualquiera de las cuatro salidas INTR de los dos 8255 hará que la terminal INTR en el microprocesador vaya a alto y solicite una interrupción.

Cuando la terminal INTR va a alto con una cadena de margaritas, el circuito no da indicación directa de cuál 8255 o cuál salida INTR se activó. La tarea de localizar cuál salida INTR se activó la realiza el procedimiento de servicio de la interrupción el cual debe "encuestar" a los dos 8255 para determinar cuál causó la interrupción. En el ejemplo 10-7 se ilustra el procedimiento de servicio de la interrupción que responde a la solicitud de interrupción de la cadena de margaritas. Este procedimiento "encuesta" a cada 8255 y a cada salida INTR para decidir cuál procedimiento de servicio de interrupción se debe utilizar.

EJEMPLO 10–7

```
                        ;Procedimiento que da servicio a la interrupción de cadena de margaritas
                        ;
= 0504                  C1      EQU  504H               ;primer 8255A
= 0604                  C2      EQU  604H               ;segundo 8255A
= 0001                  MASC1   EQU  1                  ;INTRB
= 0008                  MASC2   EQU  8                  ;INTRA

0000                    ENCUES  PROC   FAR

0000 50                         PUSH AX                 ;salvar registros
0001 52                         PUSH DX

0002 BA 0504                    MOV  DX,C1              ;direccionar puerto C
0005 EC                         IN   AL,DX
0006 A8 01                      TEST AL,MASK1
0008 75 0F                      JNZ  LEVEL_0            ;si INTRB es del primer 8255A

000A A8 08                      TEST AL,MASK2
000C 75 13                      JNZ  LEVEL_1            ;si INTRA es del primer 8255A

000E BA 0604                    MOV  DX,C2              ;direccionar puerto C
0011 EC                         IN   AL,DX
0012 A8 01                      TEST AL,MASK1
0014 75 1B                      JNZ  LEVEL_2            ;si INTRB es del segundo 8255A
0016 EB 29 9                    JMP  LEVEL_3            ;si INTRA es del segundo 8255A

0019                    ENCUES  ENDP
```

10–4 CONTROLADOR PROGRAMABLE DE INTERRUPCIONES 8259A

El controlador programable de interrupciones (PIC) 8259A agrega ocho interrupciones con vector y prioridades codificadas al microprocesador. Este controlador se puede ampliar sin más circuitos adicionales para aceptar 64 solicitudes de interrupción. Esta ampliación requiere un 8259A maestro y ocho esclavos.

FIGURA 10–14 La conexión de terminales en el controlador programable (PIC) de interrupciones 8259A.

8259A

Terminal	Señal	Señal	Terminal
11	D0	IR0	18
10	D1	IR1	19
9	D2	IR2	20
8	D3	IR3	21
7	D4	IR4	22
6	D5	IR5	23
5	D6	IR6	24
4	D7	IR7	25
27	A0		
1	$\overline{CS}$		
3	$\overline{RD}$		
2	$\overline{WR}$		
16	$\overline{SP/EN}$	CAS0	12
17	INT	CAS1	13
26	$\overline{INTA}$	CAS2	15

Descripción general del 8259A

En la figura 10-14 se ilustra el diagrama de base del 8259A, el cual es fácil de conectar con el microprocesador, porque todas las terminales, son conexiones directas excepto $\overline{CS}$ que se deben decodificar y la terminal $\overline{WR}$, que debe tener un pulso de escritura de un banco de E/S. A continuación se describen todas las terminales del 8259A:

1. D7–D0: Conexiones bidireccionales de datos. Estas terminales suelen estar conectadas con el canal superior o el inferior de datos en el microprocesador 8086, o con el canal de datos en el 8088.
2. IR7–IR0: Entradas de solicitud de interrupción. Se emplean para solicitar una interrupción y para conectar con un esclavo en un sistema que tenga 8259A múltiples.
3. $\overline{WR}$: Escribir. Una entrada conectada con la señal de habilitación estroboscópica, superior o inferior de escritura.
4. $\overline{RD}$: Leer. Entrada conectada con la señal $\overline{IORC}$.
5. INT: Interrupción. Salida conectada con la terminal INTR del microprocesador, desde el amo, y conectada con una terminal IR del amo en un esclavo.
6. $\overline{INTA}$: Reconocimiento de interrupción. Una entrada conectada con la señal $\overline{INTA}$ en el sistema. En un sistema con amo y esclavos, sólo se conecta la señal $\overline{INTA}$ del maestro.
7. A0: Dirección. Entrada que selecciona diferentes palabras de comando dentro del 8259A.
8. $\overline{CS}$: Seleccionar integrado. Se emplea para habilitar al 8259A para programa y control.
9. $\overline{SP/EN}$: Programa esclavo/habilitar acoplador. Una terminal de doble función. Cuando el 8259A está en el modo acoplador, es una salida que controla a los transreceptores en el canal de datos. Cuando el 8259A no está en este modo, esta terminal programa al controlador como amo (1) o como esclavo (0).
10. CAS2 hasta CAS0: Líneas de cascada. Se emplean como salidas del maestro hacia los esclavos para conectar en cascada a 8259A múltiples en un sistema.

Conexión de un solo 8259A

En la figura 10-15 se ilustra un solo 8259A conectado con el microprocesador 8086. En este caso, se lleva a nivel alto a la terminal $\overline{SP/EN}$ para indicar que es un maestro. Además, se debe tener en

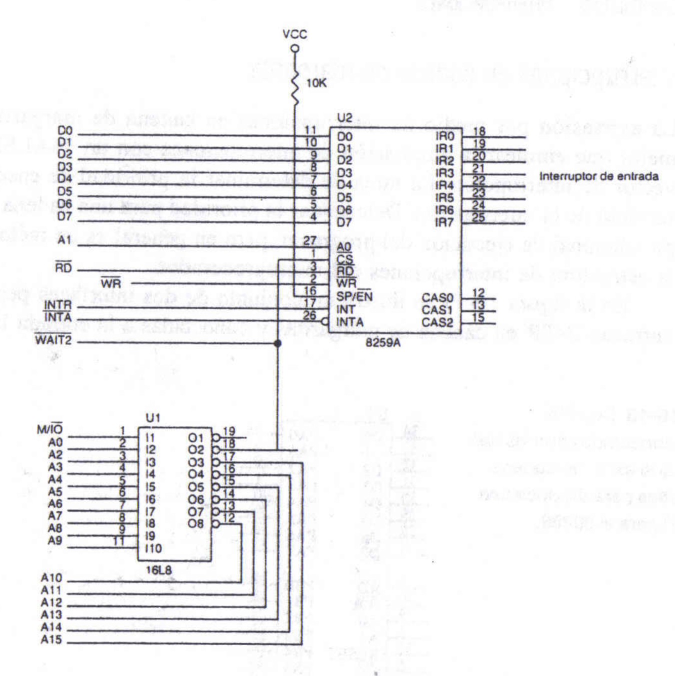


FIGURA 10–15 Un 8259A en interface con el microprocesador 8086.

cuenta que el 8259A se decodifica en los puertos 0400H y 0402H de E/S por el PAL 16L8 (no se ilustra programa). Igual que otros periféricos descritos en el capítulo anterior, el 8259A requiere dos estados de espera para funcionar bien con un 8086 de 8 MHz.

Conexión de 8259A múltiples en cascada

En la figura 10-16 se ilustran dos 8259A conectados con el microprocesador en una forma que se encuentra a menudo en la computadora AT, la cual tiene dos 8259A para interrupciones. En las computadoras XT o PC se utiliza un 8259A en los vectores de interrupción 08H–0FH. En la computadora AT se utiliza el vector de interrupción 0AH como entrada en cascada desde un segundo 8259A, ubicado en los vectores 70H hasta 77H. En el apéndice A hay una tabla en donde se enumeran las funciones de todos los vectores de interrupción utilizados en las computadoras PC, XT y AT.

En este circuito se emplean los vectores 08H–0FH y los puertos 0300H y 0302H de E/S para U1, que es el amo, y los vectores 70H–77H y los puertos 0304H y 0306H de E/S para U2, que es el esclavo. Se debe tener en cuenta que también se incluyen acopladores en el canal de datos para ilustrar el empleo de la terminal $\overline{SP}/\overline{EN}$ del 8259A. Los acopladores sólo se emplean en sistemas grandes que tienen muchos componentes conectados al canal de datos. En la práctica, rara vez se encuentran estos acopladores.

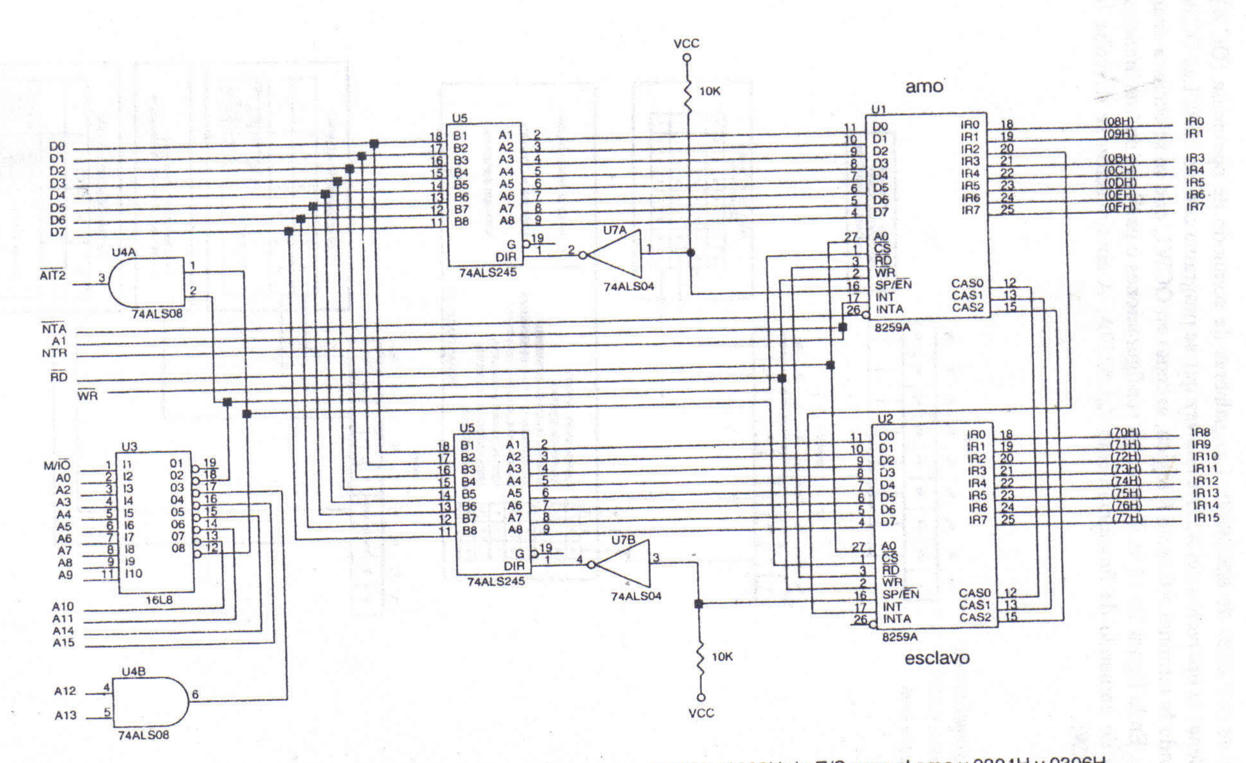


FIGURA 10–16 Dos 8259A en interface con el 8259A en los puertos 0300H y 0302H de E/S para el amo y 0304H y 0306H para el esclavo.

Programación del 8259A

El 8259A se programa con comandos de inicialización y de operación. Las palabras de comando de inicialización (ICW) se programan antes de que el 8259A pueda funcionar en el sistema y sirven para ordenar el funcionamiento básico del 8259. Las palabras de comando de operación (OCW) se programan en la operación normal y permiten el buen funcionamiento del 8259A.

Palabras de comando de inicialización. Hay cuatro palabras (ICW) de comando de inicialización para el 8259A que se seleccionan cuando la terminal A es un 1 lógico. Cuando el 8259A se alimenta por primera vez, hay que enviarle ICW1, ICW2 e ICW4. Si el 8259 se programa en el modo de cascada con ICW1, también hay que programar ICW3. Por ello, si se utiliza un solo 8259A en un sistema, se deben programar ICW1, ICW2 e ICW4. Si se utiliza el modo de cascada en un sistema, entonces hay que programar las cuatro ICW. En la figura 10-17 aparece el formato de las cuatro ICW, que se describen a continuación:

1. ICW1: Programa el funcionamiento básico del 8259A. Para programar el 8259A para trabajar con el 8086–80486, se pone un 1 lógico en el bit IC4. Los bits ADI, A7, A6 y A5 son no importa para el funcionamiento del microprocesador y sólo se aplican al 8259A cuando se utiliza con el microprocesador 8085 de 8 bits. Esta ICW selecciona funcionamiento sencillo o en cascada porque programa al bit SNGL (sencillo). Si se selecciona funcionamiento en cascada, también se debe programar ICW3. El bit LTIM determina si las entradas de solicitud de interrupción son positivas, disparadas por flanco o disparadas por nivel.
2. ICW2: Selecciona el número de vector utilizado con las entradas de solicitud de interrupción. Por ejemplo, si se decide programar a 8259A para que funcione en las localidades de vector 08H–0FH, se pone un 08H en esta palabra de comando. Asimismo, si se decide programar al 8259A para los vectores 70H–77H se pone un 70H en esta ICW.
3. ICW3: Sólo se utiliza cuando ICW1 indica que el sistema está funcionando en el modo de cascada. Esta ICW3 indica en donde está conectado el esclavo con el amo. Por ejemplo, en la figura 10-16 aparece conectado un esclavo con IR2. A fin de programar a ICW3 para esta conexión, en el amo y en el esclavo, se pone un 04H en ICW3. Supóngase que hay dos esclavos conectados con un amo por medio de IRO y de IR1. El amo se programa con una ICW3 de 03H y un esclavo se programa con una ICW3 de 01H y, el otro, con una ICW3 de 02H.
4. ICW4: Se programa para trabajar con los microprocesadores 8086–80486. Esta ICW no se programa en un sistema que trabaje con el microprocesador 8085. El bit de la extrema derecha debe ser un 1 lógico para seleccionar el funcionamiento con los microprocesadores 8086 a 80486, y los bits restantes se programan como sigue:
   a. SFNM: Selecciona el modo especial totalmente anidado de funcionamiento del 8259A, si se pone un 1 lógico en este bit. Esto permite que el maestro reconozca una solicitud de interrupción de máxima prioridad de un esclavo mientras procesa otra interrupción de un esclavo. Por lo general, sólo se procesa una solicitud de interrupción cada vez y a las demás no se las tiene en cuenta hasta que concluye el proceso.
   b. BUF Y M/S (Acoplador y amo o esclavo): Se utilizan a menudo para seleccionar el funcionamiento con o sin acoplar para el 8259A que actúe como amo o esclavo.
   c. AEOI: Selecciona el final automático o normal de la interrupción (que se describe con mayor amplitud en las Palabras de Comando de Operación). Los comandos de EOI para OCW2 se usan sólo si no se seleccciona el modo AEOI con ICW4. Si se selecciona AEOI

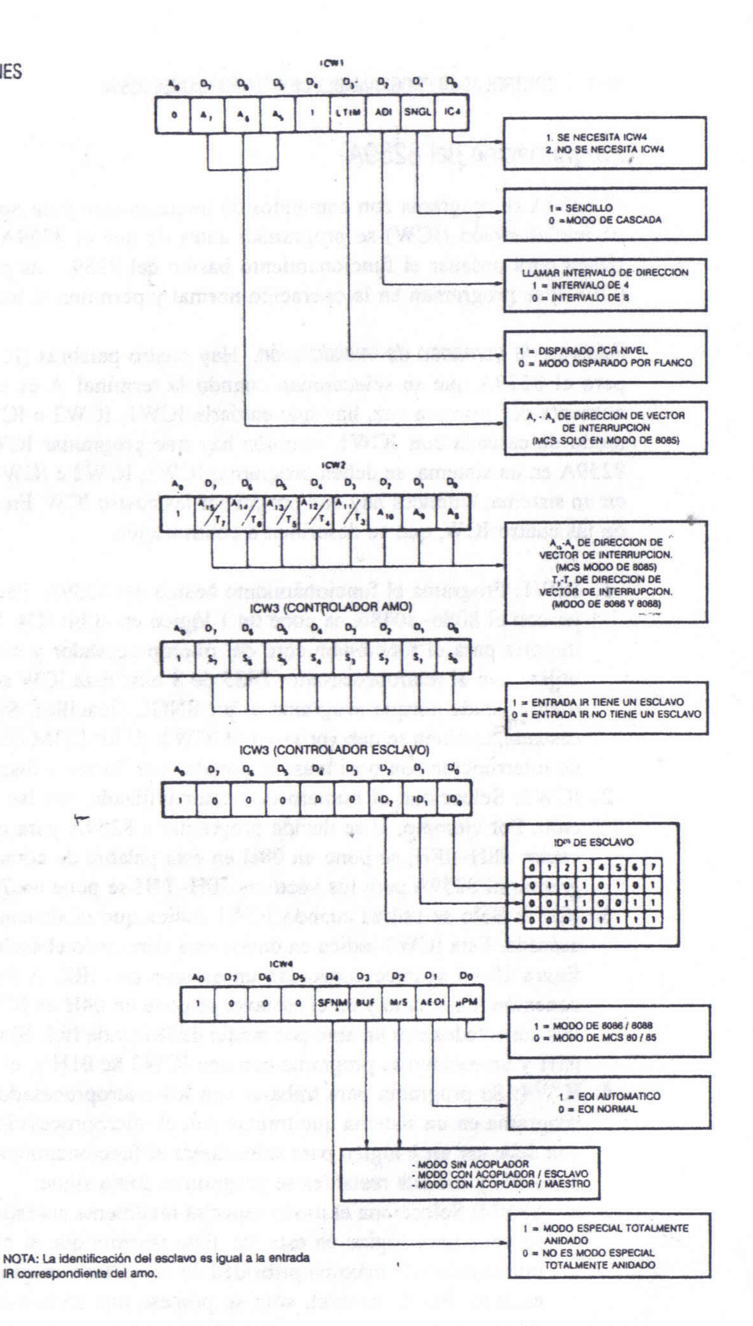


FIGURA 10–17 Las palabras de comando de inicialización (ICWS) del 8259A. (Cortesía de Intel Corporation.)

la interrupción restablece en forma automática al bit de solicitud de interrupción y no modifica la prioridad. Este es el modo de funcionamiento preferido para el 8259A y acorta el procedimiento de servicio de la interrupción.

Palabras de comando de operación. Las palabras de comando de operación (OCW) se emplean para ordenar la operación del 8259A una vez que se programa con las ICW. Las OCW se seleccionan cuando la terminal A0 es un 0 lógico, excepto en OCW1, que se selecciona cuando A0 es un 1 lógico. En la figura 10-18 se ilustran las configuraciones o patrones de bits binarios para las tres palabras de comando de funcionamiento del 8259A. A continuación se describe la función de cada OCW:

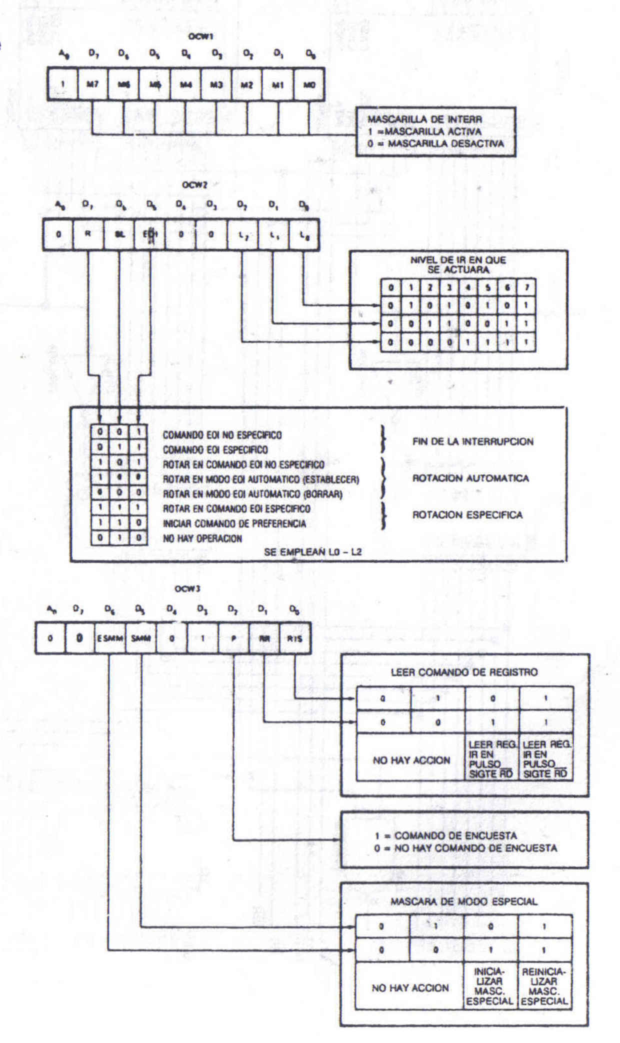


FIGURA 10–18 Las palabras de comando de operación (OCW) del 8259A. (Cortesía de Intel Corporation.)

1. OCW1: Se emplea para inicializar y leer el registro de la máscara de interrupción. Cuando se hace uno un bit de máscara, apagará (enmascara) la correspondiente entrada de interrupción. El registro de la máscara se lee cuando se lee OCW1. Debido a que el estado de la máscara se desconoce cuando se inicializa el 8259A, OCW1 se debe programar después de programar las ICW durante la inicialización.
2. OCW2: Sólo se programa cuando no se selecciona el modo AEOI para el 8259A. En este caso, la OCW selecciona la forma en que el 8259A responde a una interrupción. Esos modos son los siguientes:
   a. Final no específico de la interrupción. Un comando enviado por el procedimiento de servicio de la interrupción para señalar el final de la interrupción. El 8259A determina en forma automática cuál nivel de interrupción estaba activo y restablece al bit correspondiente del registro de estado de las interrupciones. Al restablecer al bit en el registro de estado se permite que la interrupción actúe de nuevo o que se produzca una interrupción de menor prioridad.
   b. Final específico de la interrupción. Un comando que permite restablecer una solicitud específica de interrupción. La posición exacta se determina con los bits L2–L0 de OCW2.
   c. Rotación con EOI no específico. Un comando que funciona en la misma forma exacta que el comando no específico de fin de interrupción, salvo que rota las prioridades de interrupción después de restablecer su bit en el registro de estado de las interrupciones. El valor restablecido por este comando se vuelve interrupción de mínima prioridad. Por ejemplo, si se acaba de dar servicio a IR con este comando, se convierte en la entrada de interrupción de mínima prioridad y IR5 se vuelve de máxima prioridad.
   d. Rotación con EOI automático. Un comando que selecciona un EOI automático con prioridad para la rotación. Este comando sólo se debe enviar al 8259 una vez, si se desea este modo. Si se va a apagar este modo, se debe emplear el comando para borrar.
   e. Rotar con EOI específico. Su función es similar a la de EOI específico, excepto que selecciona una prioridad de rotación.
   f. Inicializar prioridad. Permite al programador inicializar la entrada de interrupción de mínima prioridad con el empleo de los bits L2 hasta L0.
3. OWC3: Selecciona el registro que se leerá, la operación del registro de máscara especial y el comando para "encuesta". Si se selecciona encuesta, se debe activar el bit P y, luego, enviarlo al 8259A. La siguiente operación de lectura leerá la palabra de encuesta. Los tres bits de extrema derecha de la palabra de encuesta indican la solicitud de interrupción en activo con la máxima prioridad. El bit a la extrema izquierda indica si hay interrupción y hay que comprobarlo para determinar si los 3 bits de la extrema derecha contienen o no información válida.

Registro de estado. En el 8259A se pueden leer tres registros de estado, registro de solicitud de interrupción (IRR), registro de servicio (ISR) y registro de máscara de interrupción (IMR). Estos tres registros de estado se ven en la figura 10-19. El IRR es un registro de 8 bits que indica cuáles entradas de solicitud de interrupción están activadas. El ISR es un registro de 8 bits que contiene el nivel de la interrupción que se atiende. El IMR es un registro de 8 bits que contiene los bits de la máscara de interrupción y señala cuáles interrupciones están enmascaradas.

Para leer IRR e ISR se programa 0CW3 e IMR se lee por medio de OCW1. Para leer el IMR, A0 = 1 y para leer IRR o ISR, A0 = 0. Las posiciones de bits D0 y D1 de OCW3 seleccionan cuál registro (IRR o ISR) se lee cuando A0 = 0.

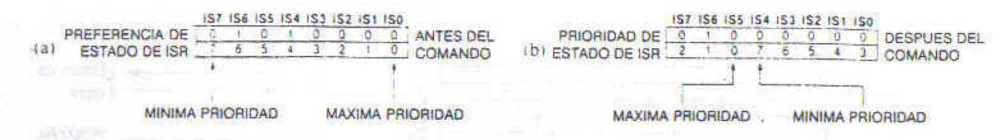


FIGURA 10–19 El registro (ISR) de 8259A en servicio. (a) Antes de aceptar IR_4 y (b) después de aceptarlo. (Cortesía de Intel Corporation.)

Ejemplo de programación del 8259A

En la figura 10-20 se ilustra el controlador de interrupciones 8259A conectado con un controlador de comunicaciones 8251A. En este circuito, hay tres terminales de salida de interrupción, del 8251A (T×RDY, R×RDY y SY/BR) que están conectadas con las entradas de solicitud de interrupción IR0, IR1 e IR2 del 8259A. La IRO ocurre cuando el transmisor está listo para enviar otro caracter. Cuando el receptor ha recibido un caracter para el 8086, IR1 está activa debido a R×RDY. La entrada IR2 se emplea para detección de rupturas. Se debe tener en cuenta que el 8251A se decodifica en los puertos 40H y 42H de 8 bits de E/S y que el 8259A se decodifica en los puertos 44H y 46H de 8 bits de E/S. Debido a que A5 no está conectada con el decodificador (puertos decodificados en forma parcial), el 8251A también funciona en los puertos 60H y 62H y el 8259A funciona en los puertos 64H y 66H. Ambos dispositivos están en interface con el canal inferior de datos.

Programa de inicialización. La primera parte del programa para este sistema debe programar al 8251A y al 8259A y, luego, habilitar la terminal INTR en el 8086 a fin de que puedan actuar las interrupciones. En el ejemplo 10-8 se lista el programa requerido para programar a ambos controladores y habilitar a INTR.

EJEMPLO 10–8 (página 1 de 3)

```
                    ;Diálogo de inicialización entre 8251A y 8259A
                    ;
= 0042              PCC_C    EQU  42H              ;registro de control en 8251A
= 0044              PIC_C1   EQU  44H              ;control 8259A, A0 = 0
= 0046              PIC_C2   EQU  46H              ;control 8259A, A0 = 1
= 0040              FORTY    EQU  40H              ;comando de inicio para 8251A
= 007B              INST1    EQU  7BH              ;palabra de modo de 8251A
= 0015              INST2    EQU  15H              ;palabra de comando de 8251A
= 001B              ICW1     EQU  1BH              ;ICW1 de 8259A
= 0080              ICW2     EQU  80H              ;ICW2 de 8259A
= 0003              ICW4     EQU  03H              ;ICW4 de 8259A
= 00F9              OCW1     EQU  0F9H             ;OCW1 de 8259A

0000                COMENSAR PROC     FAR

                    ;Inicializar el 8251A para 7 bits de datos, paridad par,
                    ;un bit de paro y un divisor de reloj de 64.
                    ;
                    ;programar 8251A
                    ;
```

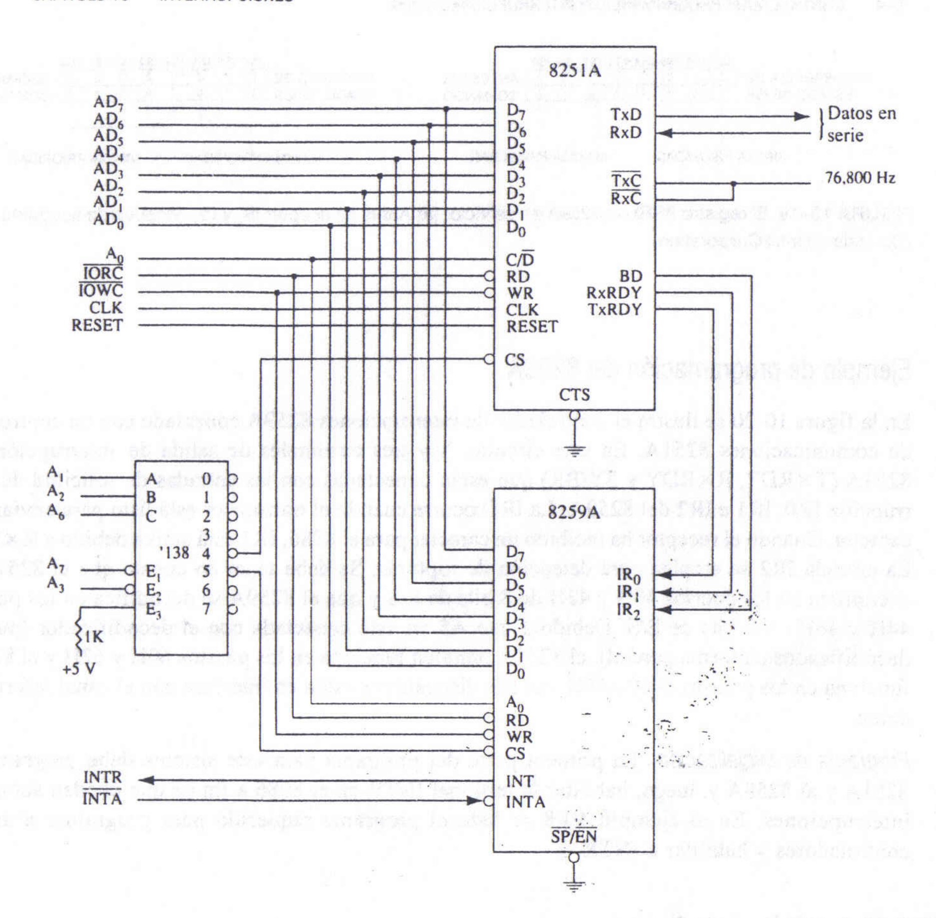


FIGURA 10–20 Ejemplo de circuito de comunicaciones con el empleo de un 8251A y de un 8259A.

EJEMPLO 10–8 (página 2 de 3)

```
0000 32 C0          XOR  AL,AL           ;reinicializar 8251A
0002 E6 42          OUT  PCC_C,AL
0004 E6 42          OUT  PCC_C,AL
0006 E6 42          OUT  PCC_C,AL
0008 B0 40          MOV  AL,FORTY
000A E6 42          OUT  PCC_C,AL

000C B0 7B          MOV  AL,INST1        ;programar modo de 8251A
000E E6 42          OUT  PCC_C,AL
0010 B0 15          MOV  AL,INST2        ;programar comando de 8251A
0012 E6 42          OUT  PCC_C,AL

               ;programar 8259A
```

EJEMPLO 10–8 (página 3 de 3)

```
0014 B0 1B              MOV  AL,ICW1        ;programar ICW1
0016 E6 44              OUT  PIC_C1,AL
0018 B0 80              MOV  AL,ICW2        ;programar ICW2
001A E6 46              OUT  PIC_C2,AL
001C B0 03              MOV  AL,ICW4        ;programar ICW4
001E E6 46              OUT  PIC_C2,AL

0020 B0 F9              MOV  AL,OCW1        ;programar OCW1
0022 E6 46              OUT  PIC_C2,AL

0024 FB                 STI                 ;habilitar INTR
0025 CB                 RET

0026          COMENSAR ENDP
```

La primera parte del procedimiento (ARRANCA) reinicializa al 8251A y programa las palabras de modo y de comando. Este modo selecciona 7 bits de datos, paridad par, un bit de paro y un divisor de reloj de 64 que ocasiona que el transmisor y el receptor transfieran datos en serie a 1200 bandios. El comando habilita al transmisor y al receptor.

La segunda parte del procedimiento programa al 8259A con sus tres ICW y su OCW. El 8259A se inicializa para funcionar en los vectores de interrupción 80H–87H y funciona con EOI automático. Las ICW habilitan a la interrupción por detección de ruptura y la interrupción del receptor, pero no la interrupión del transmisor; ésta sólo se habilita cuando hay datos que transmitir. Se habilita y deshabilita en forma periódica al transmisor durante la operación normal, con la lectura de OCW1, modificando los bits de la máscara y al volver a escribir ICW1.

Lo último que hace el procedimiento ARRANCA es habilitar la terminal INTR para que una interrupción en el receptor o por detección de ruptura, puedan ejecutarse de inmediato.

Recepción de datos desde el 8251A. Los datos recibidos por el 8251A se almacenan en una memoria FIFO hasta que el programa principal los pueda utilizar. La memoria FIFO para los datos recibidos tiene 16K bytes de longitud, por lo cual es fácil almacenar y recibir muchos caracteres antes de que se requiera la intervención del 8086 para vaciar la FIFO del receptor, la cual se almacena en el segmento extra para poder emplear instrucciones para cadenas, con el registro DI para accesar la cadena.

La recepción de datos desde el 8251A requiere dos procedimientos: uno lee el registro de datos del 8251A cada vez que la terminal R×RDY solicita una interrupción y la almacena en la FIFO; el otro, lee los datos de la FIFO desde el programa principal.

EJEMPLO 10–9

```
                ;Procedimiento principal para leer un caracter en la FIFO
                ;el caracter retorno en AL o es FEH si la FIFO está vacía.
                ;
= 0046          OCW1     EQU  46H           ;dirección de OCW1
= 00FD          MASC1    EQU  0FDH

0000            LEE      PROC   FAR

0000 53                  PUSH BX            ;salvar registros
0001 57                  PUSH DI
```

```
                        ;probar si FIFO está vacía

0002 26: 8B 3E 4002 R        MOV  DI,IOUT           ;obtener apuntador de salida
0007 26: 8B 1E 4000 R        MOV  BX,IIN            ;obtener apuntador de entrada

000C 3B DF                   CMP  BX,DI             ;probar si está vacía
000E B0 FE                   MOV  AL,0FEH           ;indicar que está vacía
0010 74 16                   JE   HECHO             ;si es que está vacía

                        ;obtener caracter de FIFO

0012 26: 8A 06               MOV  AL,ES:[DI]        ;obtener datos

                        ;modificar y salvar apuntador de salida

0015 47                      INC  DI                ;ajustar apuntador
0016 81 FF 4000 R            CMP  DI,OFFSET FIFO+16*1024
001A 26: 89 3E 4002 R        MOV  IOUT,DI
001F 76 07                   JBE  HECHO             ;dentro de límites

0021 26: C7 06 4002 R 0000 R MOV  IOUT,OFFSET FIFO  ;apuntar a arranque de FIFO

                        ;habilitar interrupción R×RDY

0028                    HECHO:

0028 50                      PUSH AX
0029 E4 46                   IN   AL,OCW1
002B 24 FD                   AND  AL,MASK1
002D E6 46                   OUT  OCW1,AL

002F 58                      POP  AX                ;restaurar registros
0030 5F                      POP  DI
0031 5B                      POP  BX
0032 CB                      RET

0033                    LEE  ENDP
```

En el ejemplo 10-9 se presenta el procedimiento utilizado para leer los datos de la FIFO en el programa principal. En este procedimiento, se supone que los apuntadores (IIN y IOUT) son inicializados en el programa de inicialización del sistema (que no se ilustra). El procedimiento (LEE) retorna con AL conteniendo un caracter leído de la FIFO. Si la FIFO está vacía, el procedimiento retorna con el caracter FEH en AL. Esto significa que no se permite a FEH como un caracter en el receptor.

EJEMPLO 10–10

```
                        ;Procedimiento de servicio de interrupción para R×RDY (IR1)
                        ;
= 0040                  DATO    EQU  40H        ;puerto de datos de 8251A
= 0046                  OCW1    EQU  46H        ;puerto OCW1 de 8259A
= 0002                  MASC2   EQU  02H        ;apagar mascarilla IR1
= 0042                  PCC_C   EQU  42H        ;puerto de comando de 8251A
= 0038                  MASC3   EQU  38H        ;mascarilla de error de 8251A

0000                    RXRDY   PROC   FAR
```

```
0000 50                          PUSH AX                   ;salvar registros
0001 53                          PUSH BX
0002 57                          PUSH DI
0003 56                          PUSH SI
0004 26: 8B 1E 4002 R            MOV  BX,IOUT              ;cargar apuntador de salida
0009 26: 8B 36 4000 R            MOV  SI,IIN               ;cargar apuntador de entrada

                          ;¿está llena FIFO?

000E 8B FE                       MOV  DI,SI
0010 46                          INC  SI
0011 81 FE 4000 R                CMP  SI,OFFSET FIFO+16*1024
0015 76 03                       JBE  NEXT
0017 BE 0000 R                   MOV  SI,OFFSET FIFO

001A                      SIGUE:

001A 3B DE                       CMP  BX,SI
001C 74 20                       JE   LLENA

                          ;probar si hay errores en 8251A

001E E4 42                       IN   AL,PCC_C             ;obtener registro de estado de 8251A
0020 24 38                       AND  AL,MASK3
0022 74 0F                       JZ   NEXT1                ;si no hay errores
0024 B0 10                       MOV  AL,10H               ;restablecer errores
0026 E6 42                       OUT  PCC_C,AL
0028 B0 3F                       MOV  AL,'?'
002A AA                          STOSB
002B 26: 89 36 4000 R            MOV  IIN,SI
0030 EB 12 90                    JMP  DONE                 ;si hay un error

                          ;leer caracter

0033                      SIG1:

0033 E4 40                       IN   AL,DATA              ;obtener datos de 8251A
0035 AA                          STOSB
0036 26: 89 36 4000 R            MOV  IIN,SI
003B EB 07 90                    JMP  DONE

                          ;si es que está llena

003E                      LLENA:

003E E4 46                       IN   AL,OCW1              ;deshabilitar IR1
0040 0C 02                       OR   AL,MASK2
0042 E6 46                       OUT  OCW1,AL

0044                      HECHO:

0044 5E                          POP  SI                   ;restaurar registros
0045 5F                          POP  DI
0046 5B                          POP  BX
0047 58                          POP  AX
0048 CF                          IRET

0049                      RXRDY  ENDP
```

En el ejemplo 10-10 se presenta el procedimiento de servicio de la interrupción R×RDY al cual se llama cada vez que el 8251A recibe un caracter para el 8086. En este ejemplo, esta interrupción emplea el vector 81H, que debe contener la dirección del procedimiento de servicio de la interrupción R×RDY. Cada vez que ocurre esta interrupción, el R×RDY lee un carácter del 8251A y lo almacena en la FIFO; si ésta se encuentra llena, se deshabilita la entrada IR1 al 8259A. Esto puede ocasionar pérdida de datos, pero cuando menos no hará que la interrupción "encime" datos válidos ya almacenados en la FIFO. Si el 8251A detecta alguna condición de error almacenan un signo ? (3FH) en la FIFO.

Transmisión de datos al 8251A. Los datos se transmiten al 8251A en una forma muy similar a la que se reciben, salvo que el procedimiento de servicio de interrupción toma los datos transmitidos de una segunda FIFO de 16K bytes.

En el ejemplo 10-11 se presenta el procedimiento que llena a la FIFO de salida. Es semejante al procedimiento que se presenta en el ejemplo 10-9, excepto que determina si la FIFO está llena en vez de vacía.

EJEMPLO 10–11

```
                        ;Procedimiento que pone datos en la FIFO de salida para transmisión
                        ;con el procedimiento de servicio de interrupción del transmisor.
                        ;AL = caracter que se va a transmitir
                        ;
= 0046                  OCW1    EQU  46H                      ;puerto OCW1 de 8259A
= 00FE                  MASC4   EQU  0FEH                     ;encender IR0

0000                    TRANS   PROC   FAR
0000 53                         PUSH BX                       ;salvar registros
0001 57                         PUSH DI
0002 56                         PUSH SI

                        ;comprobar si FIFO está llena

0003 26: 8B 36 8004 R           MOV  SI,OIN                   ;obtener apuntador de entrada
0008 26: 8B 1E 8006 R           MOV  BX,OOUT                  ;obtener apuntador de salida
000D 8B FE                      MOV  DI,SI
000F 46                         INC  SI
0010 81 FE 8004 R               CMP  SI,OFFSET OFIFO+16*1024
0014 76 03                      JBE  NEXT
0016 BE 4004 R                  MOV  SI,OFFSET OFIFO

0019                     SIGUIENTE:

0019 3B DE                      CMP  BX,SI
001B 74 06                      JE   HECHO                    ;si es que está llena,
001D AA                         STOSB
001E 26: 89 36 8004 R           MOV  OIN,SI

                         HECHO:

0023 E4 46                      IN   AL,OCW1                  ;habilitar interrupción del transmisor
0025 24 FE                      AND  AL,MASC4
0027 E6 46                      OUT  OCW1,AL

0029 5E                         POP  SI                       ;restaurar registros
002A 5F                         POP  DI
```

```
002B 5B                      POP  BX
002C CB                      RET

002D                TRANS    ENDP
```

En el ejemplo 10-12 se presenta la subrutina de servicio de la interrupción T×RDY para el transmisor de 8251A con el empleo del vector de interrupción 80H. Este procedimiento es semejante al del ejemplo 10-10 salvo que determina si FIFO está vacía en vez de llena. Se debe tener en cuenta que no se incluye el procedimiento de servicio de la interrupción para una interrupción por ruptura.

EJEMPLO 10–12

```
                    ;Procedimiento de servicio de interrupción para el transmisor 8251A
                    ;
= 0040              DATA   EQU  40H                  ;puerto de datos de 8251A
= 0001              MASK5  EQU  01H                  ;apagar IR0

002D                TXRDY  PROC   FAR

002D 50                    PUSH AX                   ;salvar registros
002E 53                    PUSH BX
002F 57                    PUSH DI
0030 26: 8B 1E 8004 R      MOV  BX,OIN               ;cargar registro de entrada
0035 26: 8B 3E 8006 R      MOV  DI,OOUT  ;cargar registro de salida

                    ;¿está vacía FIFO?

003A 3B DF                 CMP  BX,DI
003C 74 17                 JE   VACIO                ;si es que está vacía

                    ;escribir caracter

003E 26: 8A 05             MOV  AL,ES:[DI]
0041 E6 40                 OUT  DATA,AL
0043 47                    INC  DI
0044 81 FF 8004 R          CMP  DI,OFFSET OFIFO+16*1024
0048 76 03                 JBE  NEXT1
004A BF 4004 R             MOV  DI,OFFSET OFIFO

004D                NEXT1:

004D 26: 89 3E 8006 R      MOV  OOUT,DI
0052 EB 07 90              JMP  DONES

0055                VACIO:

0055 E4 46                 IN   AL,OCW1
0057 0C 01                 OR   AL,MASC5
0059 E6 46                 OUT  OCW1,AL

005B                DONES:

005B 5F                    POP  DI                   ;restaurar registros
005C 5B                    POP  BX
005D 58                    POP  AX
005E CF                    IRET

005F                TXRDY  ENDP
```

10–5 RELOJ DE TIEMPO REAL

En esta sección se presenta un reloj de tiempo real como ejemplo del empleo de una interrupción. Un reloj de tiempo real da el tiempo real, es decir, horas y minutos. En el ejemplo descrito, registra la hora en horas, minutos, segundos y 1/60 de segundo, con el empleo de cuatro localidades de memoria para llevar la hora del día en decimal en codificado binario (BCD).

En la figura 10-21 se ilustra un circuito sencillo en el cual se emplea la línea de corriente doméstica de 60 Hz para generar una señal periódica de solicitud de interrupción para la terminal de entrada de interrupción NMI. Aunque se utiliza una señal de la línea de corriente casera, cuya frecuencia puede tener ligeras variaciones, es exacta en un tiempo largo. En el circuito se utiliza una señal de la línea de 110 o 120 volts de corriente alterna (CA) acondicionada por un inversor Schmitt antes de aplicarla a la entrada de interrupción de NMI. Se debe comprobar que la tierra o neutro de la línea de corriente esté conectada a la tierra del sistema como se ilustra. La conexión a tierra de la línea es la terminal plana, grande; la clavija plana estrecha es el lado de corriente de la línea.

El programa para un reloj de tiempo real contiene un procedimiento de servicio de la interrupción, al cual se llama 60 veces por segundo y un procedimiento para actualizar el conteo ubicado en cuatro localidades de memoria. En el ejemplo 10-13 se presentan ambos procedimientos junto con los cuatro bytes de memoria utilizados para "mantener" la hora del día en BCD.

EJEMPLO 10–13

```
0000 00          HORA    DB    ?                   ;contador de 1/60 de segundo
0001 00                  DB    ?                   ;contador de segundos
0002 00                  DB    ?                   ;contador de minutos
0003 00                  DB    ?                   ;contador de horas (reloj para 24 horas)

                 ;procedimiento de interrupción de servicio para NMI

0004             TIMES   PROC  FAR

0004 50                  PUSH AX                   ;salvar registros
0005 56                  PUSH SI

0006 B4 60               MOV  AH,60H               ;cargar módulo del contador
0008 BE 0000 R           MOV  SI,OFFSET TIME       ;direccionar la hora
000B E8 0022 R           CALL UP                   ;ajustar contador de 1/60 de segundo
000E 75 0F               JNZ  DONE
0010 E8 0022 R           CALL UP                   ;ajustar contador de segundos
0013 75 0A               JZ   DONE
0015 E8 0022 R           CALL UP                   ;ajustar minutos
0018 75 05               JZ   DONE
001A B4 24               MOV  AH,24H               ;módulo 24
001C E8 0022 R           CALL UP
```

Negro 33K 74LS14 NMI
120 VAC 1K 0.1μF
Blanco

FIGURA 10–21 Conversión de la línea de corriente dómestica a una señal TTL de 60 Hz para la entrada NMI.

```
001F                 DONE:

001F 5E                    POP  SI              ;restaurar registros
0020 58                    POP  AX
0021 CF                    IRET

0022                 TIMES  ENDP

0022                 UP     PROC   NEAR

0022 2E: 8A 04             MOV  AL,CS:[SI]
0025 46                    INC  SI
0026 04 01                 ADD  AL,1            ;incrementar contador
0028 27                    DAA
0029 2E: 88 44 FF          MOV  CS:[SI-1],AL
002D 2A C4                 SUB  AL,AH
002F 75 04                 JNZ  UP1
0031 2E: 88 44 FF          MOV  CS:[SI-1],AL

0035                 UP1:

0035 C3                    RET

0036                 UP     ENDP
```

10–6 RESUMEN

1. Una interrupción es una llamada iniciada por un periférico o un programa que interrumpe el programa que se ejecuta en ese momento en cualquier punto y llama a un procedimiento, el cual es el procedimiento de servicio de la interrupción.
2. Las interrupciones son útiles cuando hay que darle servicio a un dispositivo de E/S sólo de vez en cuando y con bajas velocidades de transferencia.
3. El microprocesador tiene cinco instrucciones que se aplican a las interrupciones: BOUND, INT, INT 3, INTO e IRET. Las instrucciones INT e INT 3 se llaman procedimientos con direcciones almacenadas en un vector de interrupción, cuyo tipo se indica con la instrucción. La instrucción BOUND es condicional y utiliza el vector de interrupción 5. La instrucción INTO es condicional y sólo interrumpe un programa si está activa la bandera de sobreflujo. La instrucción IRET se emplea para retornar los procedimientos de servicio de la interrupción.
4. El microprocesador tiene tres terminales que aplican a su estructura de interrupción de periféricos: INTR, NMI e $\overline{\text{INTA}}$. Las entradas de interrupción son INTR y NMI y se emplean para solicitar interrupciones e $\overline{\text{INTA}}$ es una salida utilizada para reconocer la solicitud de interrupción INTR.
5. Las interrupciones están referidas en una tabla de vectores que ocupa las localidades de la memoria 00000H hasta 003FFH. Cada vector de interrupción tiene 4 bytes de longitud y contiene las direcciones del desplazamiento y del segmento del procedimiento de servicio de la interrupción.
6. Se utilizan dos bits de bandera con la estructura de interrupción del microprocesador, que son: trampa (TF) y habilitación de interrupciones IF. El bit de bandera IF habilita la entrada

INTR de interrupción y el bit de bandera TF hace que ocurran una interrupción después de ejecutar cada instrucción, siempre y cuando TF esté activada.

7. Los primeros 32 vectores de interrupción los reserva Intel para emplearlas en actividades definidas en el microprocesador. Los últimos 224 vectores de interrupción son para el usuario y pueden efectuar cualquier función requerida.
8. Siempre que se detecta una interrupción, ocurre lo siguiente: (1) se salvan a las banderas en la pila; (2) se desactivan los bits de las bandera IF y TF; (3) se salvan a los registros IP y CS en la pila; (4) se toma el vector de interrupción de la tabla de vectores de interrupción y se accesa el procedimiento de servicio de la interrupción por medio de la dirección del vector.
9. El trazado o paso único se logra al activar el bit de la bandera TF. Esto ocasiona una interrupción después de la ejecución de cada instrucción para depuración.
10. La entrada de la interrupción no enmascarable (NMI) llama al procedimiento cuya dirección está almacenada en el vector de interrupción 2. Esta entrada se dispara por el flanco positivo.
11. La terminal INTR no se decodifica interinamente como la terminal NMI. En vez de ello, se utiliza $\overline{\text{INTA}}$ para poner el número de vector de interrupción en las conexiones D0–D7 del canal de datos durante el pulso $\overline{\text{INTA}}$.
12. Los métodos para aplicar o poner el número de vector de interrupción en el canal de datos durante la señal $\overline{\text{INTA}}$ varían mucho. En un método, se emplean resistores para aplicar la interrupción número FFH en el canal de datos; en otro, se emplea un acoplador de tres estados para aplicar cualquier número de vector.
13. El controlador programable (PIC) de interrupciones 8259A, agrega, cuando menos, ocho entradas de interrupción al microprocesador. Si se necesitan más interrupciones, se puede conectar el controlador en cascada para producir hasta 64 entradas de interrupción.
14. La programación del 8259A es un procedimiento en dos pasos o etapas. Primero, se envía una serie de palabras de comando de inicialización (ICW) al 8259A y, después, una serie de palabras de comando de operación (OCW).
15. El 8259A contiene tres registros de estado: IMR (registro de máscara de interrupción), ISR (registro en servicio) y IRR (registro de solicitud de interrupción).
16. Un reloj de tiempo real se utiliza para dar la hora real. En la mayor parte de los casos, la hora se almacena en forma binaria o decimal codificado en binario (BCD) en cierto número de las localidades de la memoria.

10–7 CUESTIONARIO Y PROBLEMAS

1. ¿Qué se interrumpe con una interrupción?
2. Defina el término interrupción.
3. ¿Qué se llama con una interrupción?
4. ¿Por qué las interrupciones dejan tiempo libre al microprocesador?
5. Mencione las terminales para interrupción que hay en el microprocesador.
6. Mencione las cinco instrucciones de interrupción para el microprocesador.
7. ¿Qué es un vector de interrupción?
8. ¿Dónde están ubicados los vectores de interrupción en la memoria del microprocesador?
9. ¿Cuántos diferentes vectores de interrupción se encuentran en la tabla de vectores de interrupción?

10. ¿Cuántos vectores de interrupción se reserva Intel?
11. Explique cómo ocurre una interrupción tipo 0.
12. Describa el funcionamiento de la instrucción BOUND.
13. Describa el funcionamiento de la instrucción INTO.
14. ¿Cuáles localidades de memoria contiene el vector para la instrucción INT 44H?
15. Explique el funcionamiento de la instrucción IRET.
16. ¿Cuál es la finalidad del vector de interrupción número 7?
17. Enumere los acontecimientos que ocurren cuando entra en acción una interrupción.
18. Explique la finalidad de la bandera de interrupción (IF).
19. Explique la finalidad de la bandera de trampa (TF).
20. ¿Cómo se activa y desdactiva a IF?
21. ¿Cómo se activa y desactiva a TF?
22. ¿Con qué número de vector actúa en forma automática la entrada de interrupción NMI?
23. ¿Se activa la señal $\overline{INTA}$ para la terminal NMI?
24. La entrada INTR es sensible a ______________ .
25. La entrada NMI es sensible a ______________ .
26. Cuando la señal $\overline{INTA}$ se convierte en un 0 lógico, ello indica que el microprocesador espera que se coloque una interrupción número ____________ en el canal de datos en D0 hasta D7.
27. ¿Qué es una FIFO?
28. Desarrolle un circuito que coloque la interrupción 86H en el canal de datos, en respuesta a la entrada INTR.
29. Desarrolle un circuito que ponga la interrupción número CCH en el canal de datos en respuesta a la entrada INTR.
30. Explique por qué los resistores de elevación de nivel de D0–D7 hacen que el microprocesador responda al pulso $\overline{INTA}$ con un vector de interrupción tipo número FFH.
31. ¿Qué es una cadena de margaritas?
32. ¿Por qué se debe encuestar a los dispositivos para interrupción en un sistema de interrupción en cadena de margaritas?
33. ¿Qué es el 8259A?
34. ¿Cuántos 8259A se necesitan para tener 64 entradas de interrupción?
35. ¿Cuál es la finalidad de las terminales IR0 hasta IR7 en el 8259A?
36. ¿Cuándo se utilizan las terminales CAS2 hasta CAS0 en el 8259A?
37. ¿Dónde está conectada una terminal INT del esclávo en el 8259A amo en un sistema en cascada?
38. ¿Qué es una ICW?
39. ¿Qué es una OCW?
40. ¿Cuántas ICW se necesitan para programar al 8259A cuando trabaja como el único maestro en el sistema?
41. ¿Dónde se almacena el número de vector en el 8259A?
42. ¿Dónde se programa la sensibilidad de las terminales IR en el 8259A?
43. ¿Cuál es la finalidad de ICW1?
44. ¿Qué es una EOI no específica?
45. Explique la rotación en referencia al 8259A.
46. ¿Cuál es la finalidad de IRR en el 8259A?

CAPITULO 11

Acceso directo a la memoria y E/S controlado por DMA

INTRODUCCION

En capítulos anteriores se describió la atención básica y por interrupción de los periféricos de E/S. Ahora se describirá la forma final de E/S llamada acceso directo a memoria (DMA). La técnica de DMA para E/S permite el acceso directo a la memoria mientras el procesador esté deshabilitado en forma temporal. Esto permite transferir datos entre la memoria y el dispositivo E/S a una velocidad que sólo está limitada por la velocidad de los componentes de la memoria del sistema o en el controlador de DMA. La velocidad de transferencia de DMA puede estar cercana a velocidades de transferencia de 10 a 12 Mbytes con los actuales componentes de alta velocidad para la memoria RAM.

Las transferencias de DMA se emplean para muchas cosas, pero las más comunes son refrescar la DRAM, memoria de video para refrescar la pantalla y lecturas y escrituras en el sistema de memoria de disco. Las transferencias de DMA también se utilizan para transferencias de alta velocidad de memoria a memoria.

En este capítulo también se explica la operación de los sistemas de memoria de disco y los sistemas de video, que a menudo se procesan con DMA. La memoria de disco incluye almacenamiento en disquetes flexibles o duros y discos ópticos. Los sistemas de video incluyen monitores digitales y analógicos.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo el lector podrá:

1. Describir una transferencia de DMA.
2. Explicar la operación de las señales de control HOLD y HLDA para acceso directo a la memoria.
3. Explicar la operación del controlador de DMA 8237 cuando se utiliza para transferencias de DMA.
4. Programar el 8237 para transferencias de DMA.

5. Describir los diversos estándares para unidades de disco que se encuentran en computadoras personales de video que se encuentran en computadoras personales.
6. Describir los diferentes estándares para interfaces de video que se encuentran en computadoras personales.

11-1 OPERACION BASICA DE DMA

Se utilizan dos señales de control para solicitar y reconocer una transferencia de acceso directo a memoria (DMA) en un sistema basado en microprocesador. La terminal HOLD es la entrada que se utiliza para solicitar una acción de DMA y la terminal HLDA es la salida que reconoce la acción de DMA. En la figura 11-1 se ilustra la temporización típica de estas dos terminales de control de DMA.

Siempre que se lleva la entrada HOLD a un valor de 1 lógico, se solicita una acción de DMA. El microprocesador responde, en un intervalo de unos cuantos pulsos de reloj, suspendiendo la ejecución del programa y colocando los canales de direcciones, datos y de control en su estado de alta impedancia; éste, ocasiona que parezca como si se hubiera desconectado el microprocesador. Ese estado permite que los dispositivos de E/S u otros microprocesadores ganen acceso a los canales del sistema, a fin de tener acceso directo a la memoria.

Como se indica en el diagrama de temporización, se muestrea la terminal HOLD en la mitad de cualquier ciclo de reloj. Por tanto, la solicitud de DMA se puede producir en cualquier momento durante la ejecución de cualquier instrucción del microprocesador. Tan pronto como el microprocesador reconoce la solicitud, detiene la ejecución del programa y da entrada a los ciclos de DMA. Se debe tener en cuenta que la entrada HOLD tiene mayor prioridad que las entradas de interrupción INTR o NMI. Las interrupciones tienen efecto al final de una instrucción, mientras que la activación de HOLD se reconoce a la mitad de la instrucción. La única terminal del microprocesador que tiene mayor prioridad que HOLD es la terminal RESET. Se debe tener en cuenta que la entrada HOLD no debe estar activa durante la activación de RESET o no se garantiza la reinicialización.

La señal HLDA se activa para indicar que el microprocesador ha llevado a sus canales al estado de alta impedancia como se puede ver en el diagrama de temporización. Se debe tener en

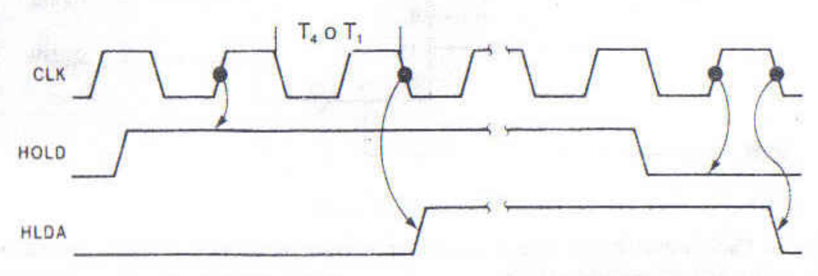


FIGURA 11-1 Temporización de HOLD y HLDA para microprocesadores 8086/8088.

cuenta que hay unos cuantos ciclos de reloj entre el momento en que cambia HOLD hasta el momento en que se activa HLDA. La salida HLDA es una señal al dispositivo solicitante externo de que el microprocesador ha dejado el control de su espacio de memoria y de E/S. Se podría llamar a la entrada HOLD una entrada de solicitud de DMA y a la salida HLDA una señal de reconocimiento de DMA.

Definiciones básicas de DMA

Los accesos directos a la memoria suelen ocurrir entre un dispositivo de E/S y la memoria sin el empleo del microprocesador. Una *lectura de DMA* transfiere datos de la memoria hacia la E/S. Una *escritura de DMA* transfiere datos de un dispositivo de E/S hacia la memoria. En ambas operaciones, la memoria y el espacio de E/S se controlan simultáneamente y eso es el porqué el sistema contiene señales separadas para control de memoria y de E/S. Esta estructura especial de los canales especiales de control del microprocesador permite las transferencias de DMA. Una lectura de DMA hace que las señales $\overline{MRDC}$ ($\overline{MEMR}$) e $\overline{IOWC}$ ($\overline{IOW}$) se activen y transfieran datos desde la memoria hasta el dispositivo de E/S. Una escritura de DMA hace que las señales $\overline{MWTC}$ ($\overline{MEMW}$) e $\overline{IORC}$ ($\overline{IOR}$) se activen. Estas señales de control del canal están disponibles en todos los microprocesadores Intel, excepto el sistema 8086/8088. Los 8086/8088 requieren su generación con un controlador del sistema o con un circuito como el que se ilustra en la figura 11-2. El controlador de DMA suministra a la memoria una dirección y la señal ($\overline{DACK}$) del controlador para seleccionar un dispositivo de E/S durante la transferencia de DMA.

La velocidad de la transferencia de datos se determina por la velocidad del dispositivo de memoria o de un controlador que, a menudo, controla las transferencias de DMA. Si la velocidad de la memoria es de 100 ns, las transferencias de DMA ocurrirán a frecuencias de hasta 1/100 ns

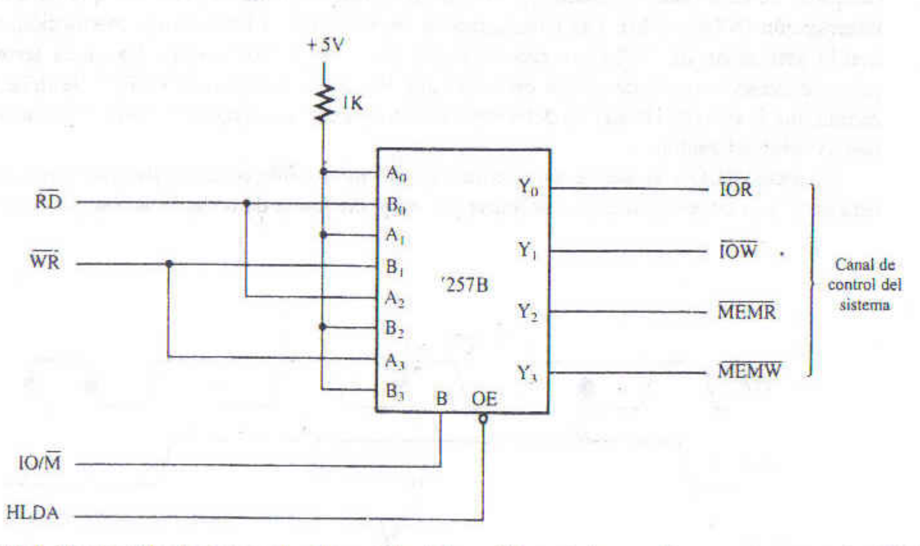


FIGURA 11–2 Circuito utilizado para producir las señales del canal de control en un sistema en que se emplea DMA en modo mínimo para el microprocesador 8088.

o sea 10 Mbytes por segundo. Si el controlador de DMA en un sistema funciona a una frecuencia máxima de 5 MHz y todavía se va a utilizar la memoria de 100 ns, la frecuencia máxima de transferencia es de 5 MHz porque el controlador de DMA es más lento que la memoria. En muchos casos, el controlador de DMA reduce la velocidad del sistema cuando ocurren las transferencias de DMA.

11–2 EL CONTROLADOR 8237 DE DMA

El controlador de DMA 8237 suministra a la memoria y al E/S señales de control e información y direccionamiento a la memoria durante la transferencia de DMA. El 8237 es, en realidad, un microprocesador de propósito especial cuya labor es la transferencia de datos a alta velocidad entre la memoria y el espacio de E/S. En la figura 11-3 se muestra el diagrama de base y el diagrama a bloques del controlador programable de DMA 8237. Aunque este controlador quizá no aparezca como componente discreto en sistemas modernos basados en microprocesador, se emplea en los conjuntos de controladores integrados (chip-set) que hay en los sistemas más nuevos. Aunque el conjunto de controladores integrados no se describe por su complejidad, el conjunto 82357 (controlador integrado de periféricos o ISP) integra dos controladores de DMA que se programan en la misma forma exacta que el 8237. El ISP también incluye un par de controladores programables de interrupciones 8259A para el sistema.

El 8237 tiene cuatro canales y es compatible con los microprocesadores 8086 y 8088. El 8237 puede expandirse para incluir cualquier número de entradas de canal de DMA, aunque parece ser que cuatro canales son adecuados para muchos sistemas pequeños. El 8237 puede

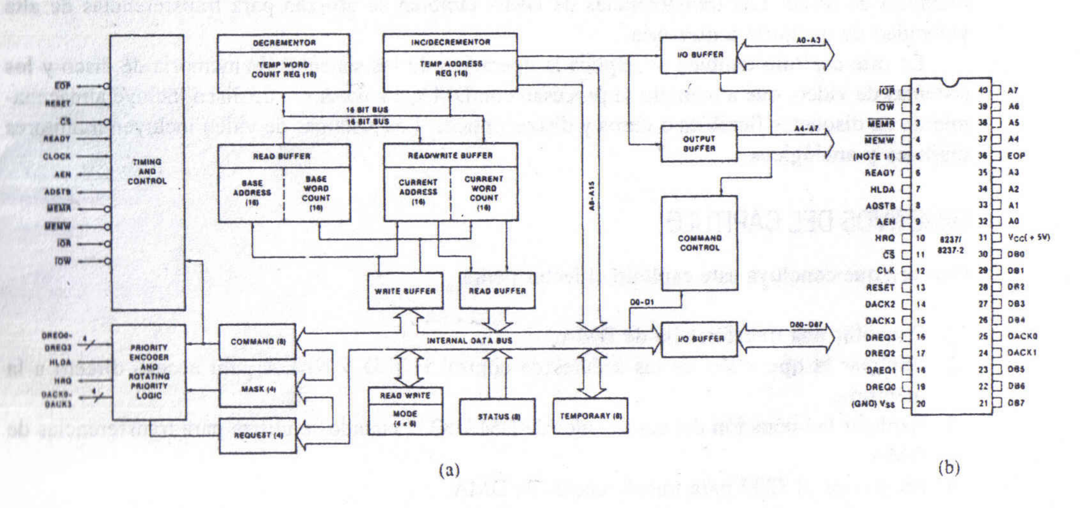


FIGURA 11–3 El controlador programable 8237A-5 para DMA. (a) Diagrama de bloque. (b) Conexiones de terminales. (Cortesía de Intel Corporation.)

efectuar transferencias de DMA a velocidades de hasta 1.6 Mbytes por segundo. Cada canal puede direccionar a una sección completa de 64 Kbytes de la memoria y puede transferir hasta 64 Kbytes con una sola programación.

Definiciones de terminales

1. CLK (reloj) está conectada con la señal de reloj del sistema siempre y cuando ésta sea de 5 MHz o menos. En el sistema de 8086 y 8088, hay que invertir el reloj para tener buen funcionamiento del 8237.
2. $\overline{CS}$ (selección de integrado) selecciona al 8237 durante la programación. Esta terminal, por lo general, está conectada con la salida de un decodificador, el cual no utiliza la señal de control IO/$\overline{M}$(M/$\overline{IO}$) del 8086 y 8088 porque contiene las nuevas señales de control de la memoria y E/S: ($\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ e $\overline{IOW}$).
3. RESET (Inicializar) borra los registros de comandos, estado, solicitud y temporales. También borra el primero y último flip-flop e inicializa el registro de enmascaramiento. Esta entrada reinicializa al 8237 deshabilitándolo hasta que se programa.
4. READY (Listo) hace que el 8237 inserte estados de espera para los componentes de memoria o E/S más lentos.
5. HLDA (reconoce cesión) le señala al 8327 que el microprocesador ha cedido el control de los canales de dirección, datos y control.
6. $DREQ_3$, DRQ_0: (solicitud de DMA) son entradas que se emplean para solicitar transferencias de DMA para cada uno de los cuatro canales de DMA. Debido a que la polaridad de estas entradas es programable, son entradas activas en alto o bajo.
7. DB_7 hasta DB_0 (canal de datos). Son las conexiones al canal de datos del microprocesador y se emplean durante la programación del controlador de DMA.
8. $\overline{IOR}$ (leer E/S) es una terminal bidireccional utilizada durante la programación del controlador de DMA.
9. $\overline{IOW}$(escribir en E/S) es una terminal bidireccional utilizada durante la programación y durante un ciclo de escritura de DMA.
10. $\overline{EOP}$ (final del proceso) es una señal bidireccional que se emplea como entrada para terminar un proceso de DMA o como salida para señalar el final de una transferencia de DMA. Esta entrada se emplea a menudo para interrumpir una transferencia de DMA al final de un ciclo de DMA.
11. A_3 hasta A_0 (direcciones) son terminales que se utilizan para seleccionar un registro interno durante la programación y también son parte de la dirección de DMA durante una acción de DMA.
12. A_7 hasta A_4 (direcciones) son salidas que proporcionan parte de la dirección de transferencia de DMA durante una acción de DMA.
13. HRQ (retener solicitud), es una salida que se conecta a la entrada HOLD del microprocesador a fin de solicitar una transferencia de DMA.
14. $DACK_3$-$DACK_0$ (reconocimiento de canal de DMA) son salidas que aceptan o reconocen una solicitud de un canal de DMA. Estas salidas se pueden programar como activas en alto o activas bajo. Las salidas DACK se emplean a menudo para seleccionar el dispositivo de E/S controlado por DMA durante la transferencia de DMA.
15. AEN (habilitación de dirección) habilita el registro de direccionamiento DMA conectado a las terminales DB_7–DB_0 del 8237. También se utiliza para deshabilitar cualquier acoplamiento a microprocesador del sistema conectadas con el microprocesador.

16. ADSTB (señal estroboscópica de habilitación de dirección) funciona como ALE, salvo que se emplea en el controlador de DMA para capturar los bits de dirección A_{15}–A_8 durante la transferencia de DMA.
17. $\overline{MEMR}$ (lectura a memoria) es una salida que hace que se lean datos de la memoria durante un ciclo de lectura de DMA.
18. $\overline{MEMW}$ (escritura a memoria) es una salida que hace que se escriban datos en la memoria durante un ciclo de escritura de DMA.

Registros internos

1. *Registro de dirección actual (CAR)* contiene la dirección de memoria de 16 bits utilizada para la transferencia de DMA. Cada canal tiene su propio registro CAR para ello. Cuando se transfiere un bit de datos durante una operación de DMA, se incrementa o decrementa el CAR, según se programó.
2. *Registro de contador. (CWCR)* programa a un canal para el número de bits (hasta 64) transferidos durante una acción de DMA. El número cargado en este registro es uno menos que el número de bits transferidos. Por ejemplo, si se carga un 10 en el CWCR, entonces se transfieren 11 bits durante la acción de DMA.
3. *Dirección base (BA) y contador de palabras base (BWC)*, son registros que se emplean para seleccionar la autoinicialización automática de un canal. En el modo de autoinicialización, estos registros se emplean para volver a cargar los registros CAR y CWCR después de que ha concluido la acción de DMA. Estos permite que el mismo conteo y dirección se utilicen para transferir datos de la misma área de memoria.
4. *Registro de comandos (CR)* programa la operación del controlador 8237. La figura 11-4 ilustra la función del registro de comandos.

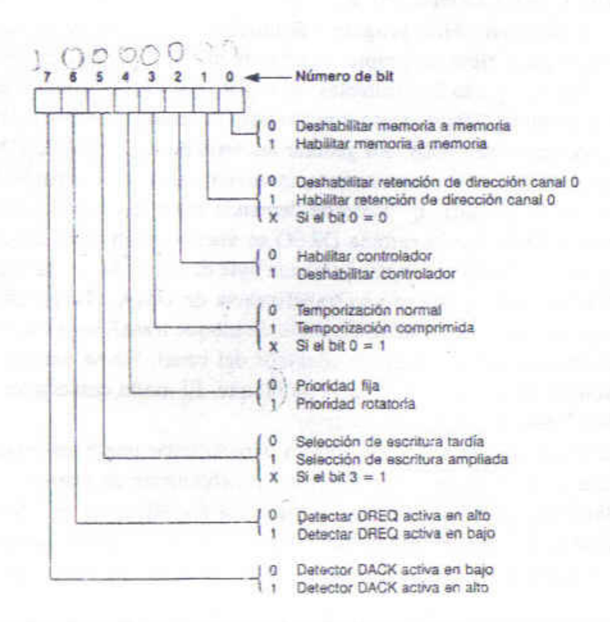


FIGURA 11–4 Registro de comandos en el 8237A-5. (Cortesía de Intel Corporation.)

En el registro de comandos se utiliza la posición de bit 0 para seleccionar el modo de transferencia de memoria a memoria de DMA. En esas transferencias se emplea el canal 0 de DMA para la dirección fuente y el canal de DMA 1 para la dirección de destino. (Esto es semejante al funcionamiento de una instrucción MOVSB). Se lee un byte en la dirección accesada por el canal 0 y se salva en el 8237 en un registro temporal. Luego, el 8237 inicia un ciclo de escritura a la memoria, en donde el contenido del registro temporal se escribe a la dirección seleccionada por el canal 1 de DMA. El número de bytes transferidos lo determina el registro contador del canal 1.

El bit de habilitación de retención de la dirección del canal 0 (posición de bit 1) programa al canal 0 para transferencias de memoria a memoria. Por ejemplo, supóngase que se debe llenar una zona de la memoria con datos; el canal 0 puede tener la misma dirección mientras el canal 1 cambia para hacer transferencia de memoria a memoria. Así se copia el contenido de la dirección a del canal 0, en un bloque de memoria al que apunta y accesa el canal 1.

El bit de habilitación y deshabilitación controlador (posición de bit 2) enciende y apaga todo el controlador. El bit normal y compresión (posición de bit 3) determina si un ciclo de DMA consta de 2 periodos de reloj (comprimido) o de 4 (normal). La posición de bit 5 se emplea en la temporización normal para extender el pulso de escritura por lo cual aparece un pulso de reloj más temprano en la temporización para dispositivos de E/S que requieren un pulso de escritura más ancho.

La posición de bit 4 selecciona la prioridad de las entradas DRWQ de los 4 canales de DMA. En el esquema de prioridad fija, el canal 0 tiene máxima prioridad y el canal 3 la mínima. En el esquema de prioridad rotatoria, el canal al cual se acaba de dar servicio, toma la mínima. Por ejemplo, si el canal 2 acaba de tener acceso a una transferencia de DMA, toma la mínima prioridad y el canal 3 toma la máxima prioridad. El esquema rotatorio es un intento por dar igual prioridad a todos los canales.

Los dos bits restantes (posiciones de bit 6 y 7), programan las polaridades de las entradas DRWQ y de las salidas DACK.

5. *Registro de modo (MR),* programa de modo de operación de un canal. Se debe tener en cuenta que cada canal tiene su propio registro de modo (figura 11-5), seleccionado por las posiciones de bit 1 y 0. Los bits restantes del registro de modo seleccionan operación, autoinicialización, incremento/decremento y modo para el canal. Las operaciones de verificación generan las direcciones de DMA, sin generar las señales de control de DMA memoria y de E/S.

   Los modos de funcionamiento, incluyen: modo de demanda, modo único, modo de bloque y modo cascada. El modo de demanda transfiere datos hasta que entra una señal $\overline{EOP}$ externa o hasta que la entrada DREQ se vuelve inactiva. El modo único o sencillo libera la señal HOLD después de transferir cada byte de datos. Si se mantiene activa la terminal DREQ el 8237 solicita de nuevo una transferencia de DMA a través de la línea DRQ a la entrada HOLD del microprocesador. El modo de bloque transfiere en forma automática el número de bytes indicado por el registro contador del canal. No se necesita mantener activa a DREQ durante el modo de transferencia de bloque. El modo cascada se emplea cuando hay más de un 8237 que trabaje en un sistema.

6. *Registro de solicitud (RR)* se emplea para solicitar una transferencia de DMA por programa. (figura 11-6). Esto es muy útil en las transferencias de memoria a memoria, cuando no está disponible una señal externa para iniciar la transferencia de DMA.

7. *Activar o desactivar registro de enmascaramiento (MRSR),* activa o desactiva el registro de enmascaramiento de cada canal como se ilustra en la figura 11-7. Si se activa el enmasca-

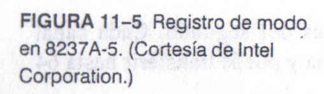


FIGURA 11–5 Registro de modo en 8237A-5. (Cortesía de Intel Corporation.)

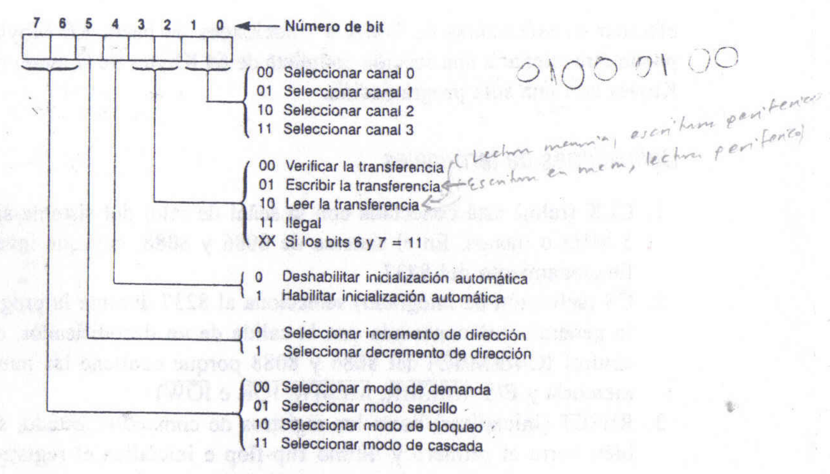


FIGURA 11–6 Registro de solicitud de 8237A-5. (Cortesía Intel Corporation.)

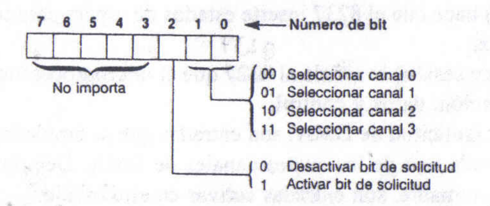
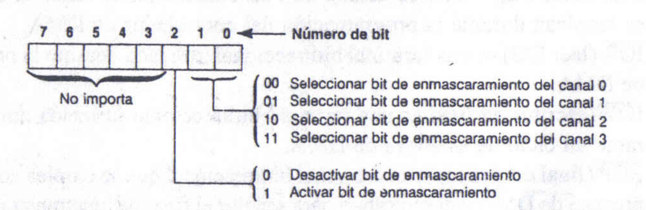


FIGURA 11–7 Modo de habilitar/deshabilitar registro de enmascaramiento del 8237A-5. (Cortesía de Intel Corporation.)



ramiento, se deshabilita el canal. Recuerde que la señal RESET activa todos los enmascaramientos deshabilitando todos los canales.

8. *Registro de enmascaramiento(OMSR)* (figura 11-8) habilita o deshabilita todos los enmascaramientos con un comando, en lugar del enmascaramiento individual de cada canal como con MRSR.
9. *Registro de estado (SR)* muestra el estado de cada canal de DMA (figura 11-9) los bits TC indican si el canal ha llegado a su cuenta terminal (transferencia de todos sus bytes). Cuando se llega a la cuenta final o terminal, se termina la transferencia de DMA para casi todos los modos de funcionamiento. Los bits de solicitud indican si está activa la entrada DREQ para un canal específico.

Comando de programación. Se emplean tres comandos para controlar la operación del 8237. Estos comandos no tienen un patrón de bits, como ocurre con los diversos registros de control



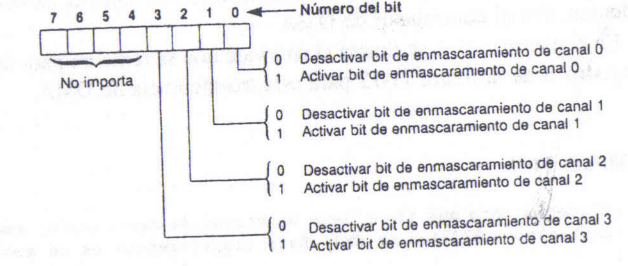


FIGURA 11–8 Regsistro de enmascaramiento de 8237A-5. (Cortesía de Intel Corporation.)

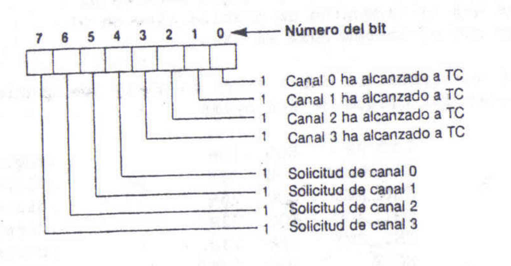


FIGURA 11–9 Registro de estado de 8237A-5. (Cortesía Intel Corporation.)

FIGURA 11–10 Asignaciones de puertos de comando y control. en 8237A-5. (Cortesía de Intel Corporation.)

Señales						Operación
A3	A2	A1	A0	$\overline{IOR}$	$\overline{IOW}$	
1	0	0	0	0	1	Leer registro de estado
1	0	0	0	1	0	Escribir registro de comandos
1	0	0	1	0	1	Ilegal
1	0	0	1	1	0	Escribir registro de solicitudes
1	0	1	0	0	1	Ilegal
1	0	1	0	1	0	Escribir un bit del registro de enmascaramiento
1	0	1	1	0	1	Ilegal
1	0	1	1	1	0	Escribir registro de modo
1	1	0	0	0	1	Ilegal
1	1	0	0	1	0	Borrar flip-flop de apuntador de bytes
1	1	0	1	0	1	Leer registro temporal
1	1	0	1	1	0	Borrado maestro
1	1	1	0	0	1	Ilegal
1	1	1	0	1	0	Borrar registro de enmascaramiento
1	1	1	1	0	1	Ilegal
1	1	1	1	1	0	Escribir todos los bits del registro de enmascaramiento

dentro del 8237. Una salida sencilla al número de puerto debido, habilita el comando de programación. En la figura 11-10 se ilustran las asignaciones de puertos de E/S que accesan a todos los registros y comandos de programación.

A continuación se explican las funciones de los comandos de programación:

1. *Borrar el primero/último flip-flop* borra el primero/último (F/L) flip-flop dentro del 8237. Ese flip-flop selecciona cuál byte (alto o bajo) se lee o escribe en los registros de dirección y de conteo actuales. Si F/L = 0, se selecciona el byte de orden bajo; si es F/L = 1 se selecciona el de orden alto. Cualquier lectura o escritura en la dirección o el registro de conteo, produce inversión del flip-flop F/L.

2. *Inicio maestro.* Actúa en la misma forma exacta que la señal RESET para el 8237. Este comando, igual que la señal RESET deshabilita a todos los canales.
3. *Desactivar registro de enmascaramiento* habilita a los cuatro canales de DMA.

Programación de los registros de dirección y de conteo. En la figura 11-11 se ilustran las localidades de los puertos de E/S para programar los registros de dirección y de contadores de cada canal. Se verá que el estado del flip-flop F/L determina si se programa el byte menos significativo (LSB) o el más significativo (MSB). Si se desconoce el estado del flip-flop F/L, los registros de dirección y contador se podrían programar en forma incorrecta.

Hay cuatro pasos para programar el 8237: (1) se desactiva el flip-flop F/L mediante el comando para desactivarlo; (2) se deshabilita el canal; (3) se programa primero el LSB y luego el MSB de la dirección; (4) se programan el LSB y el MSB del contador. Una vez efectuado lo anterior, el canal está programado y listo para usarse. Se necesita programación adicional para seleccionar el modo de funcionamiento antes de que se habilite y ponga a funcionar el canal.

Canal	Registro	Operación	Señales							Flip-flop interno	Canal de datos, DB0 hasta DB7
			$\overline{CS}$	$\overline{IOR}$	$\overline{IOW}$	A3	A2	A1	A0		
0	Dirección base y actual	Escribir	0	1	0	0	0	0	0	0	A0-A7
			0	1	0	0	0	0	0	1	A8-A15
	Dirección actual	Leer	0	0	1	0	0	0	0	0	A0-A7
			0	0	1	0	0	0	0	1	A8-A15
	Contador de palabras base y actual	Escribir	0	1	0	0	0	0	1	0	W0-W7
			0	1	0	0	0	0	1	1	W8-W15
	Contador de palabras actuales	Leer	0	0	1	0	0	0	1	0	W)0-W7
			0	0	1	0	0	0	1	1	W8-W15
1	Dirección base y actual	Escribir	0	1	0	0	0	1	0	0	A0-A7
			0	1	0	0	0	1	0	1	A8-A15
	Dirección actual	Leer	0	0	1	0	0	1	0	0	A0-A7
			0	0	1	0	0	1	0	1	A8-A15
	Contador de palabras base y actual	Escribir	0	1	0	0	0	1	1	0	W0-W7
			0	1	0	0	0	1	1	1	W8-W15
	Contador de palabras actuales	Leer	0	0	1	0	0	1	1	0	W)0-W7
			0	0	1	0	0	1	1	1	W8-W15
2	Dirección base y actual	Escribir	0	1	0	0	1	0	0	0	A0-A7
			0	1	0	0	1	0	0	1	A8-A15
	Dirección actual	Leer	0	0	1	0	1	0	0	0	A0-A7
			0	0	1	0	1	0	0	1	A8-A15
	Contador de palabras	Escribir	0	1	0	0	1	0	1	0	W0-W7
			0	1	0	0	1	0	1	1	W8-W15
	Contador de palabras actuales	Leer	0	0	1	0	1	0	1	0	W)0-W7
			0	0	1	0	1	0	1	1	W8-W15
3	Dirección base y actual	Escribir	0	1	0	0	1	1	0	0	A0-A7
			0	1	0	0	1	1	0	1	A8-A15
	Dirección actual	Leer	0	0	1	0	1	1	0	0	A0-A7
			0	0	1	0	1	1	0	1	A8-A15
	Contador de palabras base y actual	Escribir	0	1	0	0	1	1	1	0	W0-W7
			0	1	0	0	1	1	1	1	W8-W15
	Contador de palabras actuales	Leer	0	0	1	0	1	1	1	0	W)0-W7
			0	0	1	0	1	1	1	1	W8-W15

FIGURA 11–11 Direcciones de los puertos de canal E/S de DMA en el 8237A-5. (Cortesía de Intel Corporation.)

Conexión del 8237 con el microprocesador 8088

En la figura 11-12 se ilustra un sistema basado en el 8088 que contiene al controlador de DMA 8237.

La salida de habilitación de dirección (AEN) del 8237 controla a las terminales de salida de los registros y las salidas del multiplexor 74LS257 (E). Durante la operación normal del 8088 (AEN = 0), los registros A y C y el multiplexor (E) proporcionan los bits A_{19}–A_{16} y A_7–A_0 del canal de direcciones. El multiplexor produce las señales de control del sistema, siempre y cuando se encuentre bajo el control del 8088. Durante una acción de DMA (AEN = 1), se deshabilitan los registros A y C junto con el multiplexor (E). Ahora los registros D y B producen los bits A_{19}–A_{16} y A_{15}–A_8 de dirección. Los bits A_7–A_0 del canal de direcciones los proporciona en forma directa el 8237 y contienen una parte de la dirección de transferencia de DMA. El controlador de DMA también produce las señales de control $\overline{MEMR}$, $\overline{MEMW}$, $\overline{IOR}$ e $\overline{IOW}$.

La señal de habilitación estroboscópica de salida (ADSTB) del 8237 habilita los bites de salidad de la dirección (A_{15}–A_8)en el registro D durante la acción de DMA, por lo cual toda la dirección de transferencia de DMA queda disponible en el canal de direcciones. Los bits A_{19}–A_{16} de la dirección los produce el registro B, el cual se debe programar con estos cuatro bits de dirección antes de habilitar el controlador para la transferencia de DMA. El funcionamiento del 8237 para DMA está limitado a una transferencia de no más de 64 Kbytes, dentro de la misma sección de 64 Kbytes de la memoria.

El decodificador (F) selecciona al 8237 para su programación y también 4 bits del registro (B) de 4 bits para los cuatro bits más altos de la dirección. El decodificador de este sistema habilita al 8237 para las direcciones XX70H-XX7FH del puerto de E/S y el registro (B) de E/S para los puertos XX10H-XX1FH. Se debe tener en cuenta que la salida del decodificador se combina con la señal $\overline{IOW}$ para generar una señal de reloj activa en alto para el registro (B).

Durante la operación normal del 8088, se deshabilitan el controlador de DMA y los circuitos integrados B y D. Durante una acción de DMA, se deshabilitan los circuitos A, C y E, para que el 8237 pueda tomar el control del sistema a través de los canales de direcciones, datos y control.

En una computadora personal con memoria paginada, los dos controladores de DMA se programan como sigue: canales 0 a 3: puertos 0000H-000FH; canales 4 a 7; 00C0H-00DF.

Se debe tener en cuenta que el segundo controlador sólo se programa en las direcciones par, por lo cual la dirección base y actual del canal 4 se programan en el puerto 00C0H de E/S y el contador base y actual del canal 4 se programan en el puerto 00C2H. El registro de páginas, que contiene los bits de dirección A_{23}–A_{16} de la dirección de DMA se encuentra en los puertos de E/S 0087H (canal 0), 0083H (canal 1), 0081H (canal 2), 0082H (canal 3, no hay canal 4) 008BH (canal 5), 0089H (canal 6) y 008AH (canal 7). El registro de páginas funciona como el registro de multiplexión de la dirección que se describe en otros ejemplos de la presente obra.

Transferencia de memoria a memoria con el 8237

La transferencia de memoria a memoria es mucho más potente que incluso la instrucción MOVSB con repetición automática. Para el 8088 opera a 4.2 microsegundos por bit, el 8237 sólo necesita 2.0 microsegundos por bit. Es casi el doble de rápido que un programa de transferencia de datos.

Muestra de transferencia de DMA de memoria a memoria. Supóngase que se va a transferir el contenido de las localidades de memoria 10000H hasta 13FFH a las localidades de memoria

14000H hasta 17FFFH. Se logra con una instrucción repetida de mover cadena o, con mucha más velocidad, con el controlador de DMA.

En el ejemplo 11-1 se ilustra el software que se requiere para inicializar al 8237 y programar el registro B de la figura 11-12 para esta transferencia de DMA.

EJEMPLO 11–1

```
                ;Procedimiento que transfiere un bloque de datos con el empleo del
                ;controlador 8237A-5 de DMA. Esta transferencia es de memoria a memoria.
                ;
                ;Se supone que la dirección de fuente está en SI
                ;Se supone que la dirección de destino está en DI
                ;Se supone que el conteo está en CX
                ;
                ;Se supone que todos los datos están en el ES que contiene
                ;la localidad del bloque de 64K bytes
                ;
= 0010                  LATCHB    EQU  10H        ;registro (B)
= 007C                  CLEAR_FL  EQU  7CH        ;flip-flop F/L
= 0070                  CH0_ADD   EQU  70H        ;dirección de CH0
= 0072                  CH1_ADD   EQU  72H        ;dirección de CH1
= 0073                  CH1_CNT   EQU  73H        ;conteo en CH1
= 007B                  MODE      EQU  7BH        ;modo
= 0078                  CMMD      EQU  78H        ;comando
= 007F                  MASKS     EQU  7FH        ;mascarilla
= 0079                  REQ       EQU  79H        ;registro de solicitudes
= 0078                  STATUS    EQU  78H        ;registro de estado

0000                    TRANSFER  PROC FAR

0000 50                           PUSH AX

0001 8C C0                        MOV  AX,ES      ;registro del programa (B)
0003 8A C4                        MOV  AL,AH
0005 D0 E8                        SHR  AL,1
0007 D0 E8                        SHR  AL,1
0009 D0 E8                        SHR  AL,1
000B D0 E8                        SHR  AL,1
000D E6 10                        OUT  LATCHB,AL

000F E6 7C                        OUT  CLEAR_FL,AL ;borrar F/L
0011 8B C6                        MOV  AX,SI      ;programa fuente
0013 E6 70                        OUT  CH0_ADD,AL
0015 8A C4                        MOV  AL,AH
0017 E6 70                        OUT  CH0_ADD,AL

0019 8B C7                        MOV  AX,DI      ;programa destino
001B E6 72                        OUT  CH1_ADD,AL
001D 8A C4                        MOV  AL,AH
001F E6 72                        OUT  CH1_ADD,AL

0021 8B C1                        MOV  AX,CX      ;contador del programa
0023 48                           DEC  AX         ;ajustar contador
0024 E6 73                        OUT  CH1_CNT,AL
0026 8A C4                        MOV  AL,AH
0028 E6 73                        OUT  CH1_CNT,AL
002A B0 88                        MOV  AL,88H     ;programar el modo
002C E6 7B                        OUT  MODE,AL
```

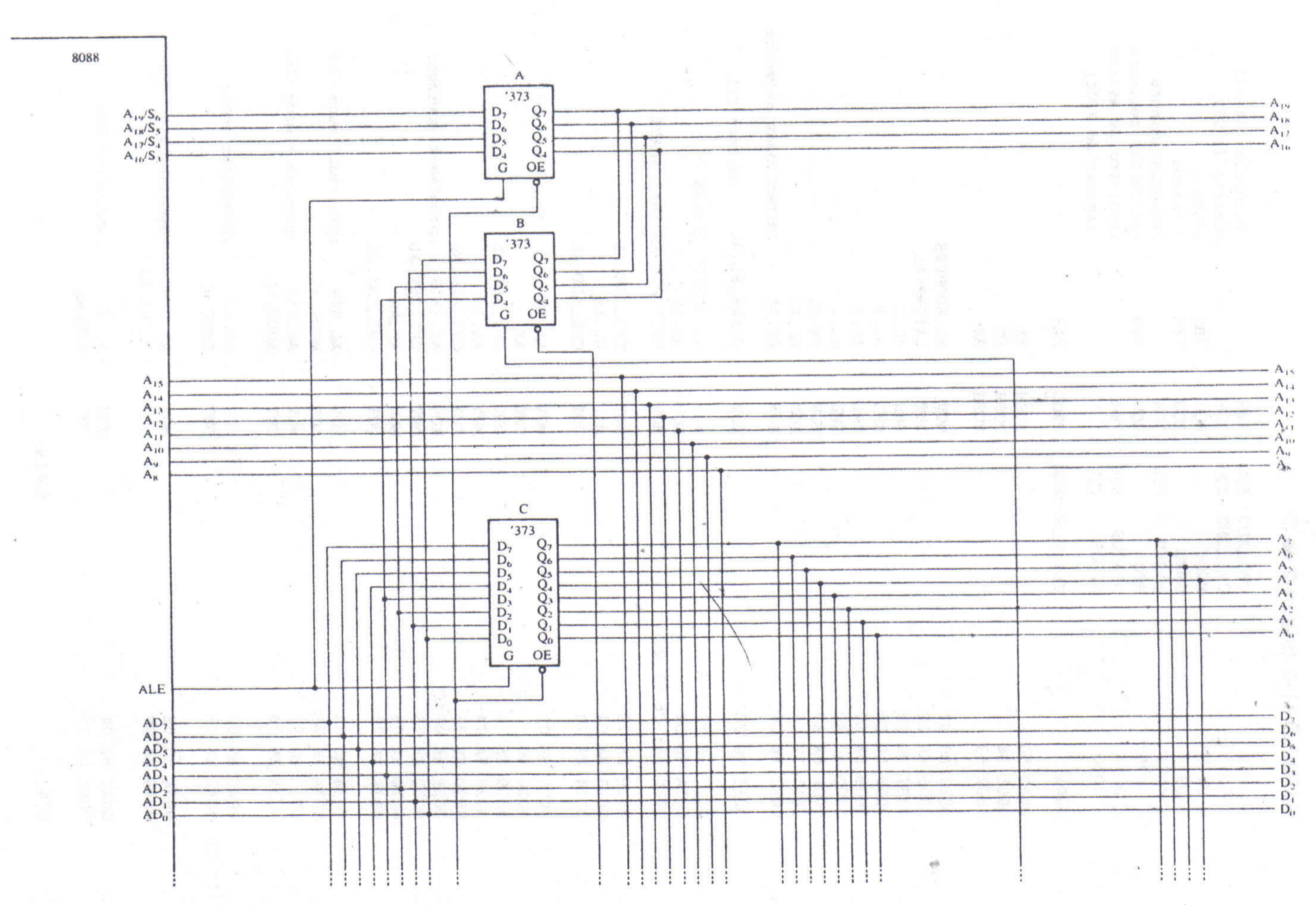
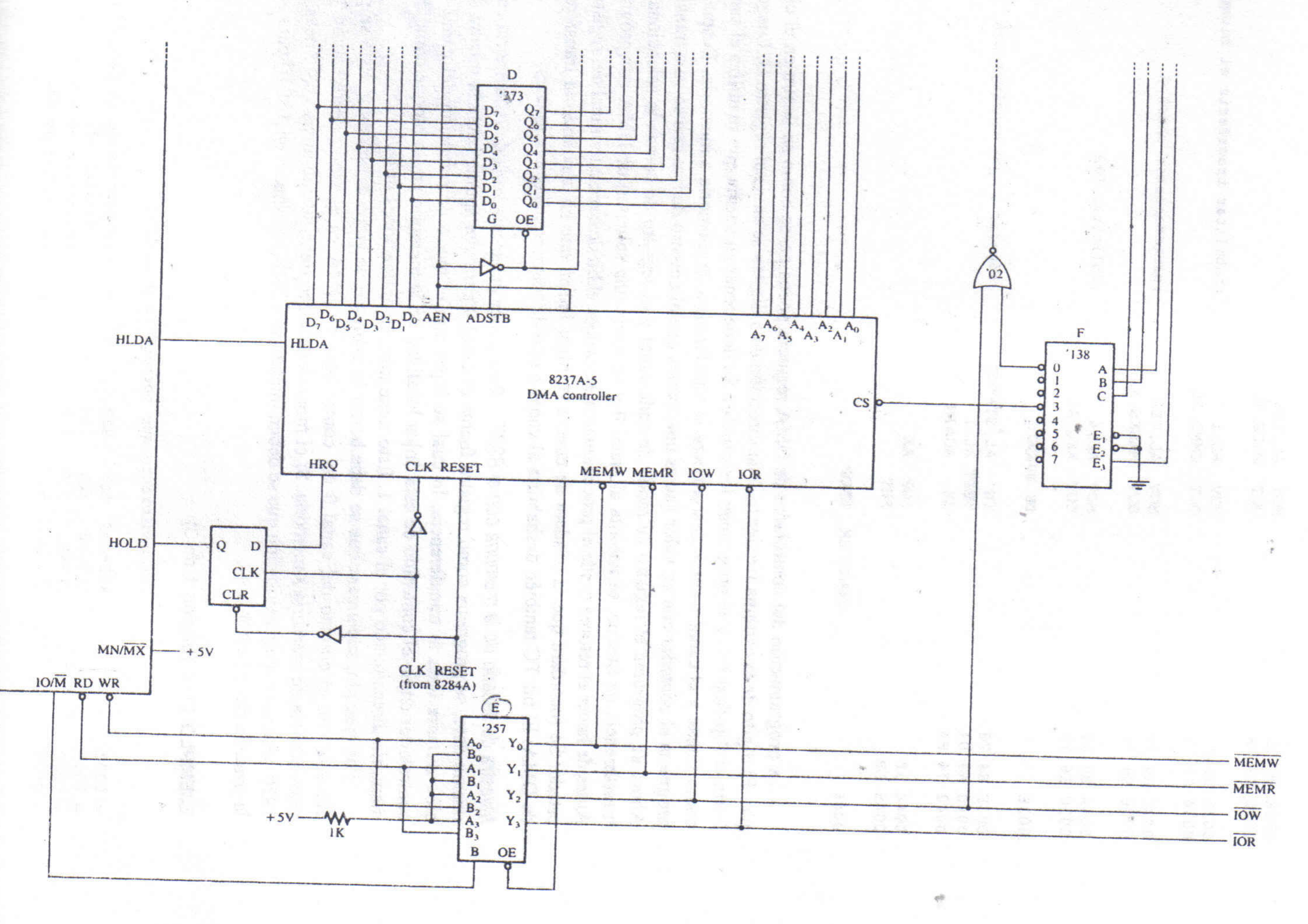


FIGURA 11–12 Sistema completo en modo mínimo de DMA para 8088.


8237A-5
DMA controller
'373
'138
'02
'257
HLDA
HOLD
MN/$\overline{MX}$
+5V
IO/$\overline{M}$ RD WR
CLK RESET
(from 8284A)
HRQ
AEN
ADSTB
CS
MEMW MEMR IOW IOR
$\overline{MEMW}$
$\overline{MEMR}$
$\overline{IOW}$
$\overline{IOR}$
1K

```
002E B0 85                  MOV  AL,85H
0030 E6 7B                  OUT  MODE,AL

0032 B0 01                  MOV  AL,1            ;habilitar transferir el bloque
0034 E6 78                  OUT  CMMD,AL

0036 B0 0E                  MOV  AL,0EH          ;desenmascarar canal 0
0038 E6 7F                  OUT  MASKS,AL

003A B0 04                  MOV  AL,4            ;solicitar DMA
003C E6 79                  OUT  REQ,AL

003E                        DE NUEVO:

003E E4 78                  IN   AL,STATUS       ;esperar a que concluya DMA
0040 A8 01                  TEST AL,1
0042 74 FA                  JZ   AGAIN

0044 58                     POP  AX
0045 CB                     RET

0046              TRANSFER  ENDP
```

La programación del controlador de DMA requiere pocos pasos como se ilustra en el ejemplo. El dígito de la extrema izquierda de la dirección de 5 dígitos se envía al registro B. Luego, se borra el flip-flop F/L y se programan los canales. Se debe tener en cuenta que se utiliza el canal 0 como fuente y el canal 1 como destino para la transferencia de memoria a memoria. Después se programa el contador con un valor que es uno menos que el número de bits que se va a transferir. Ahora se programa el registro de modo de cada canal y el registro de comando selecciona una transferencia en bloque, se habilita al canal 0 y se inicia una solicitud de DMA por programa. Antes de hacer el retorno desde el procedimiento, se prueba el bit de cuenta terminal del registro de estado. Se recordará que la bandera de cuenta terminal indica que ha concluido la transferencia de DMA. El bit TC también deshabilita al canal e impide transferencias adicionales.

Muestra de llenado de la memoria con el 8237. Para poder llenar una zona de la memoria con un mismo dato, se programa como registro fuente el canal 0 para que apunte hacia la misma dirección durante toda la transferencia, lo cual se logra con el modo de retención del canal 0. El controlador copia el contenido de esta única localidad de la memoria en un bloque completo de memoria direccionado por el canal 1. Esto tiene muchas aplicaciones útiles.

Por ejemplo, supóngase que se debe borrar la memoria de video del monitor. Esto se puede efectuar con el empleo del canal 0 del controlador de DMA en el modo de retención y una transferencia de memoria a memoria. Si el monitor de video tiene 80 columnas y 24 líneas, tiene 1920 posiciones para exhibición que se deben inicializar a 20H (un espacio en ASCII) para apagar la pantalla del monitor.

EJEMPLO 11–2 (página 1 de 3)

```
                ;Procedimiento que borrará la pantalla.
                ;
= 0002          ADDRESS   EQU  02H        ;Dirección de la pantalla
= 0010          LATCHB    EQU  10H        ;registro (B)
= 007C          CLEAR_FL  EQU  7CH        ;flip-flop de F/L
= 0070          CH0_ADD   EQU  70H        ;Dirección de CH0
```

EJEMPLO 11–2 (página 2 de 3)

```
= 0072              CH1_ADD  EQU    72H             ;Dirección de CH1
= 0073              CH1_CNT  EQU    73H             ;Conteo de CH1
= 007B              MODE            EQU     7BH     ;modo
= 0078              CMMD            EQU     78H     ;comando
= 007F              MASKS    EQU    7FH             ;enmascaramiento
= 0079              REQ             EQU     79H     ;solicitar registro
= 0078              STATUS   EQU    78H             ;registro de estado
= 0020              SPACE    EQU    ' '             ;espacio de ASCII

0000                CLEAR_SCREEN    PROC    FAR

0000 50                             PUSH    AX
0001 06                             PUSH    ES
0002 53                             PUSH    BX

0003 B0 02                          MOV     AL,ADDRESS
0005 E6 10                          OUT     LATCHB,AL
0007 D0 E0                          SHL     AL,1
0009 D0 E0                          SHL     AL,1
000B D0 E0                          SHL     AL,1
000D D0 E0                          SHL     AL,1
000F 8A E0                          MOV     AH,AL
0011 32 C0                          XOR     AL,AL
0013 8E C0                          MOV     ES,AX      ;direccionar segmento

0015 E6 7C                          OUT     CLEAR_FL,AL    ;borrar F/L

0017 B8 0014 R                      MOV     AX,OFFSET SCREEN
001A 8B D8                          MOV     BX,AX
001C 26: C6 07 20                   MOV     BYTE PTR ES:[BX],SPACE

0020 E6 70                          OUT     CH0_ADD,AL
0022 8A C4                          MOV     AL,AH
0024 E6 70                          OUT     CH0_ADD,AL

0026 8B C3                          MOV     AX,BX
0028 40                             INC     AX
0029 E6 72                          OUT     CH1_ADD,AL
002B 8A C4                          MOV     AL,AH
002D E6 72                          OUT     CH1_ADD,AL
002F B8 077E                        MOV     AX,1918    ;programar contador
0032 E6 73                          OUT     CH1_CNT,AL
0034 8A C4                          MOV     AL,AH
0036 E6 73                          OUT     CH1_CNT,AL

0038 B0 88                          MOV     AL,88H     ;programar modo CH0
003A E6 7B                          OUT     MODE,AL
003C B0 85                          MOV     AL,85H     ;programar modo CH1
003E E6 7B                          OUT     MODE,AL

0040 B0 03                          MOV     AL,3       ;programar copia
0042 E6 78                          OUT     CMMD,AL

0044 B0 0E                          MOV     AL,0EH     ;desenmascarar CH0
0046 E6 7F                          OUT     MASKS,AL

0048 B0 04                          MOV     AL,4       ;solicitar DMA
004A E6 79                          OUT     REQ,AL

004C                   AGAIN:
```

EJEMPLO 11–2 (página 3 de 3)

```
004C E4 78                          IN      AL,STATUS  ;esperar a que concluya
004E A8 01                          TEST    AL,1
0050 74 FA                          JZ      DE NUEVO

0052 5B                             POP     BX
0053 07                             POP     ES
0054 58                             POP     AX
0055 CB                             RET

0056             CLEAR_SCREEN       ENDP
```

En el ejemplo 11-2 se muestra un procedimiento que borra la memoria de video. En este caso, se conoce la dirección de la memoria y el número de bits que hay en la pantalla. Se debe tener en cuenta que este procedimiento es semejante al ejemplo 11-1, excepto que se programa a través del registro de comandos por lo cual se retiene la dirección del canal 0.

Interface con una impresora procesada por DMA

En la figura 11-13 se ilustra el circuito agregado al de la figura 11-12 para tener una interface con una impresora procesada por DMA. El registro se emplea para capturar los datos que le envían a la impresora durante la transferencia de DMA. El pulso de escritura pasado a través del registro durante la acción de DMA también genera la señal de habilitación de datos ($\overline{DS}$) a la impresora por medio del multivibrador monoestable. La impresora envía la señal $\overline{ACK}$ cada vez que está lista para datos adicionales. En este circuito, se emplea $\overline{ACK}$ para solicitar una acción de DMA por medio del flip-flop.

Se debe tener en cuenta que para seleccionar el dispositivo E/S no se decodifica la dirección en el canal de direcciones. Durante la transferencia de DMA el canal de direcciones contiene la dirección de la memoria y no puede contener la dirección del puerto de E/S. En lugar de la dirección de E/S la salida $DACK_3$ del 8237 selecciona el registro pasando el pulso de escritura a través de una compuerta OR.

El programa que controla esta interface es sencillo, porque sólo se programan la dirección de los datos y el número de caracteres que se van a imprimir. Una vez programados, se habilita el canal y la acción de DMA transfiere un bit cada vez que la interface recibe la señal $\overline{ACK}$ desde la impresora.

El procedimiento que imprime los datos desde el segmento de datos actual se ilustra en el ejemplo 11-3. Este procedimiento programa al 8237, pero, en realidad, no imprime nada. La impresión se logra con el controlador de DMA y la interface de la impresora.

EJEMPLO 11–3 (página 1 de 3)

```
                 ;Procedimiento que imprime datos a partir de la localidad
                 ;direccionada por DS:BX. El número de caracteres que se imprimen
                 ;está en CX cuando se llama a este procedimiento.
                 ;

= 0010           LATCHB     EQU   10H          ;registro (B)
= 007C           CLEAR_FL   EQU   7CH          ;flip-flop F/L
= 0076           CH3_ADD    EQU   76H          ;dirección de CH3
= 0077           CH3_CNT    EQU   77H          ;contador de CH3
```

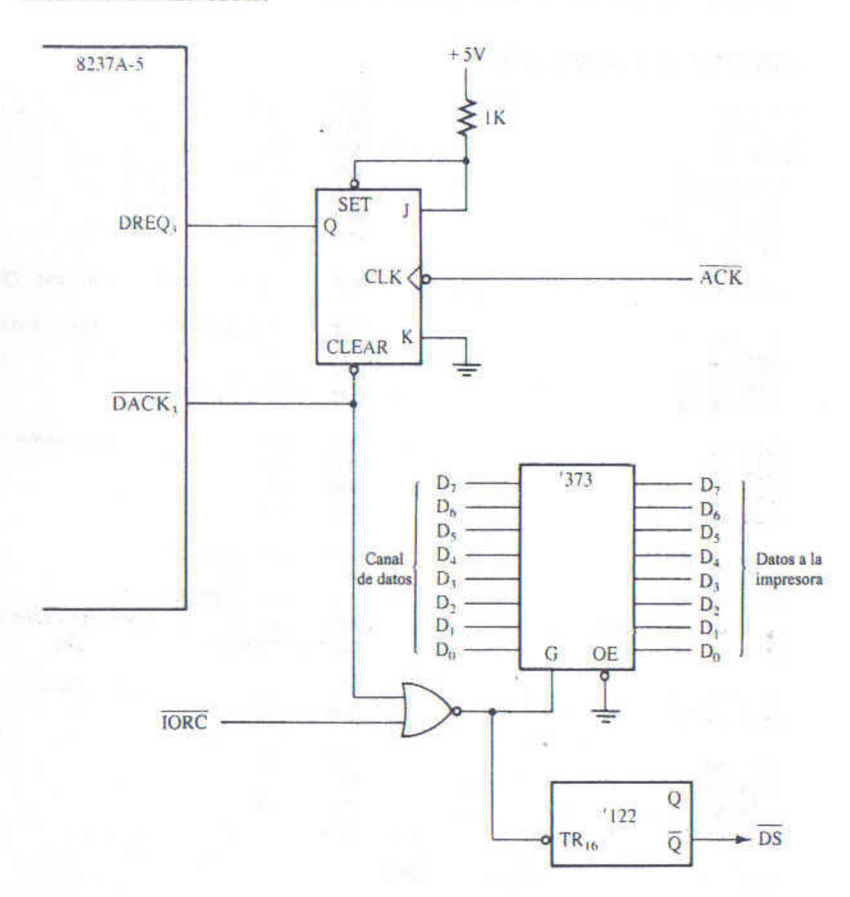


FIGURA 11–13 Interface con impresora para proceso de DMA.

EJEMPLO 11–3 (página 2 de 3)

```
= 007B              MODE          EQU    7BH     ;modo
= 0078              CMMD          EQU    78H     ;comando
= 007F              MASKS   EQU   7FH            ;mascarilla
= 0079              REQ           EQU    79H     ;registro de solicitud

0000                PRINT   PROC  FAR

0000 50                           PUSH   AX
0001 51                           PUSH   CX
0002 53                           PUSH   BX

0003 8C D8                        MOV    AX,DS   ;programar A19-A16
0005 B1 04                        MOV    CL,4
0007 D3 EB                        SHR    BX,CL
0009 03 C3                        ADD    AX,BX
000B B1 0C                        MOV    CL,12
000D 50                           PUSH   AX
000E D3 E8                        SHR    AX,CL
```

EJEMPLO 11–3 (página 3 de 3)

```
0010 E6 10                        OUT    LATCHB,AL
0012 58                           POP    AX
0013 5B                           POP    BX
0014 B1 04                        MOV    CL,4
0016 D3 E0                        SHL    AX,CL
0018 83 E3 0F                     AND    BX,15
001B 03 C3                        ADD    AX,BX

001D E6 7C                        OUT    CLEAR_FL,AL      ;borrar F/L

001F E6 76                        OUT    CH3_ADD,AL       ;programar
address
0021 8A C4                        MOV    AL,AH
0023 E6 76                        OUT    CH3_ADD,AL

0025 58                           POP    AX               ;programa contador
0026 50                           PUSH   AX
0027 53                           PUSH   BX
0028 48                           DEC    AX

0029 E6 77                        OUT    CH3_CNT,AL
002B 8A C4                        MOV    AL,AH
002D E6 77                        OUT    CH3_CNT,AL
002F B0 00                        MOV    AL,0       ;habilita controlador
0031 E6 78                        OUT    CMMD,AL

0033 B0 07                        MOV    AL,7       ;desenmascarilla CH3
0035 E6 7F                        OUT    MASKS,AL

0037 5B                           POP    BX
0038 59                           POP    CX
0039 58                           POP    AX
003A CB                           RET

003B              PRINT      ENDP
```

Se necesita un procedimiento secundario para determinar si ha concluido la acción de DMA. En el ejemplo 11-4 se presenta el procedimiento secundario que prueba el controlador de DMA para determinar si ha concluido la transferencia. Se llama al procedimiento PRUEB_I antes de programar el controlador de DMA para ver si se terminó la transferencia anterior.

EJEMPLO 11–4

```
                  ;Procedimiento para comprobar si ha concluido una acción de DMA.
                  ;
= 0078            STATUS     EQU   78H              ;estado

0000              TEST_P     PROC  NEAR

0000 E4 78                   IN    AL,STATUS        ;probar CH3
0002 A8 08                   TEST  AL,8
0004 74 FA                   JZ    TEST_P
0006 C3                      RET

0007              TEST_P     ENDP
```

Los datos a imprimir se pueden tener en dos áreas de la memoria si primero se carga un área de la memoria con los datos que se van a imprimir. Luego, se llama al procedimiento IMPRIME para que empiece a imprimir los datos del área 1 de la memoria. Debido a que se requiere muy poco tiempo para programar el controlador de DMA, se puede llenar una segunda área de la memoria con nuevos datos para la impresora, mientras los datos del área 1 se imprimen con la interface de la impresora y el controlador de DMA. Este proceso se repite hasta que se han impreso todos los datos.

11–3 FUNCIONAMIENTO DEL CANAL (BUS) COMPARTIDO

Los sistemas complejos de computadoras de la actualidad tienen tantas tareas que efectuar, que algunos sistemas emplean más de un microprocesador para efectuar al trabajo. Esto se llama *sistema de multiprocesamiento*. A veces, también se le llama *sistema distribuido*. A un sistema que lleva a cabo más de una tarea, se le llama sistema *multitarea*. En sistemas que tienen más de un microprocesador se debe crear y emplear algún método de control. En un ambiente distribuido, multiprocesamiento multitareas, cada microprocesador accesa 2 canales: (1) el *canal local* y (2) el *canal remoto o compartido*.

En esta sección se describe la operación del canal compartido para los microprocesadores 8086 y 8088 con el empleo del árbitro de canal 8289. En el 80286 se emplea el árbitro 82289 y en los 80386 y 80486, se emplea el 82389. Estos sistemas son mucho más difíciles y complejos para describirlos en este capítulo, pero la terminología y funcionamiento son las mismas que para 8086 y 8088.

El canal local está conectado con la memoria y con los dispositivos de E/S a que accesa directamente un solo microprocesador sin ningún protocolo o reglas de acceso especiales. El canal remoto (compartido) tiene memoria y dispositivos de E/S a los cuales tiene acceso cualquier microprocesador del sistema. En la figura 11-14 se ilustra esta situación con unos cuantos microprocesadores.

Tipos de canales

El canal local es en el que "reside" sólo un microprocesador. Este canal contiene la memoria residente o local y los dispositivos de E/S. Todos los microprocesadores descritos hasta ahora en este libro se consideran sistemas de canal local. La memoria y E/S locales se accesan con el microprocesador que esté conectado directamente con ellos.

Un canal compartido es el que esté conectado con todos los microprocesadores del sistema. Este canal se emplea para intercambiar datos entre los microprocesadores del sistema. El acceso al canal compartido se controla con alguna forma de arbitraje que permite a un solo microprocesador acceso al espacio del canal compartido del sistema.

En la figura 11-15 se ilustra un microprocesador 8088 conectado como maestro de canal remoto. El término *maestro del canal* se aplica a cualquier dispositivo (microprocesador u otro) que pueda controlar un canal que contenga memoria y E/S. El controlador de DMA 8237, ya descrito, es un ejemplo de un maestro de canal remoto. El 8237 tiene acceso a la memoria del

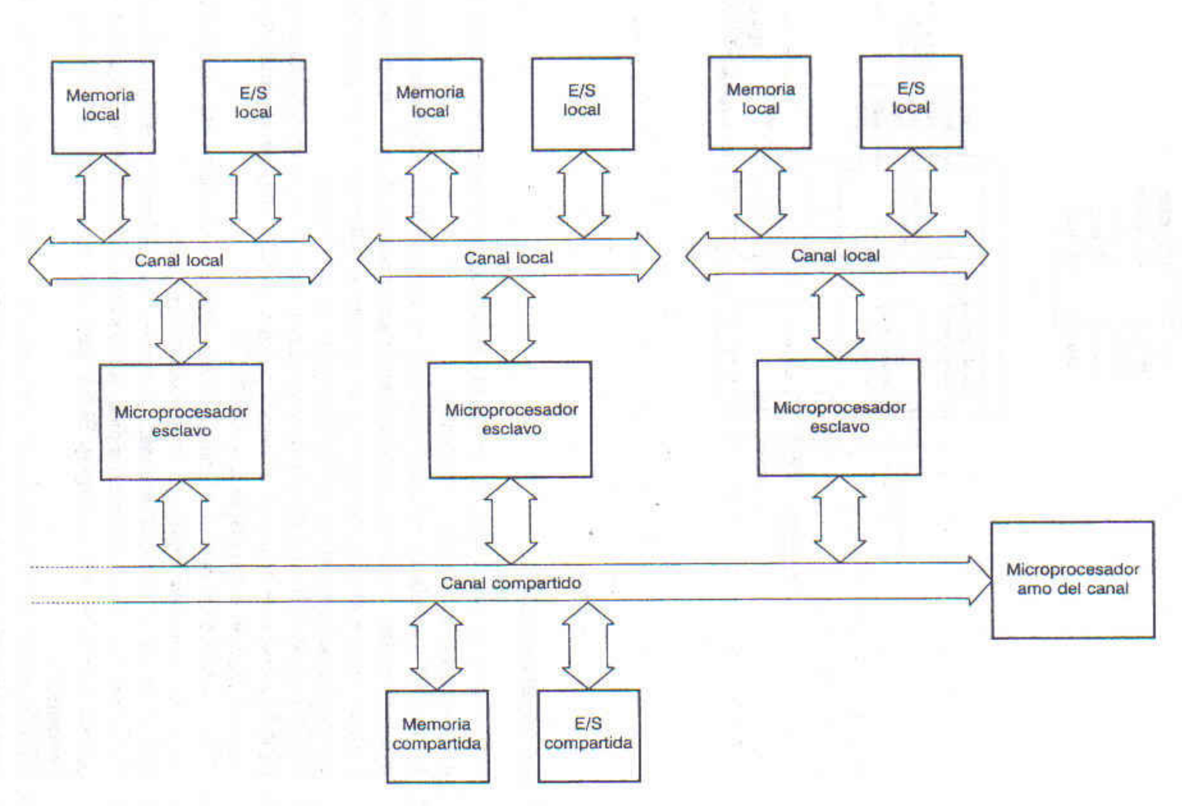


FIGURA 11–14 Diagrama de bloques que ilustra los canales compartidos y los locales.

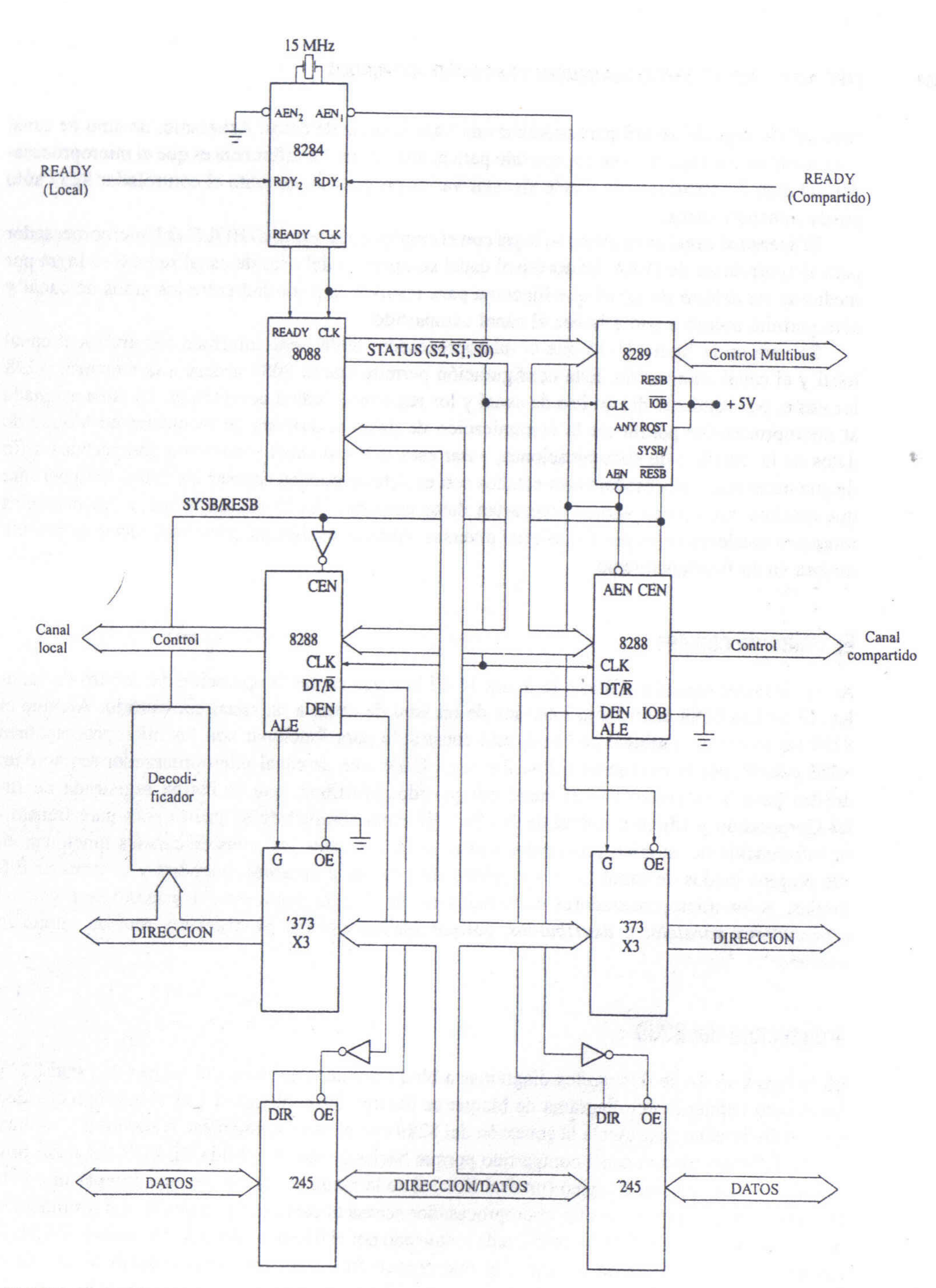


FIGURA 11-15 El 8088 cuando funciona en el modo remoto y se ilustran las conexiones de canales local y remoto.

sistema y al espacio de E/S para producir una transferencia de datos. Asimismo, un amo de canal remoto obtiene acceso al canal compartido para el mismo fin. La diferencia es que el microprocesador del amo del canal remoto puede ejecutar varios programas, en tanto el controlador 8237 sólo puede transferir datos.

El acceso al canal compartido se logra con el empleo de la terminal HOLD del microprocesador para el controlador de DMA. El acceso al canal compartido del amo de canal remoto se logra por medio de un *árbitro de canal* que funciona para resolver la prioridad entre los amos de canal y sólo permite acceso a uno a la vez al canal compartido.

Se verá en la figura 11-15 que el microprocesador 8088 tiene interface con ambos el canal local y el canal compartido. Esta configuración permite que el 8088 accese a la memoria y E/S locales o, por conducto del árbitro de canal y los registros al canal compartido. La tarea asignada al microprocesador podría ser la comunicación de datos y, después de recolectar un bloque de datos de la interface de comunicaciones, pasar esos datos al canal y memoria compartidos a fin de que otros microprocesadores conectados con el sistema puedan accesar los datos. Esto permite que muchos microprocesadores compartan datos comunes. En la misma forma, a los múltiples microprocesadores se les pueden asignar diversas tareas en el sistema, para tener una muy notable mejora en su funcionamiento.

El árbitro de canales

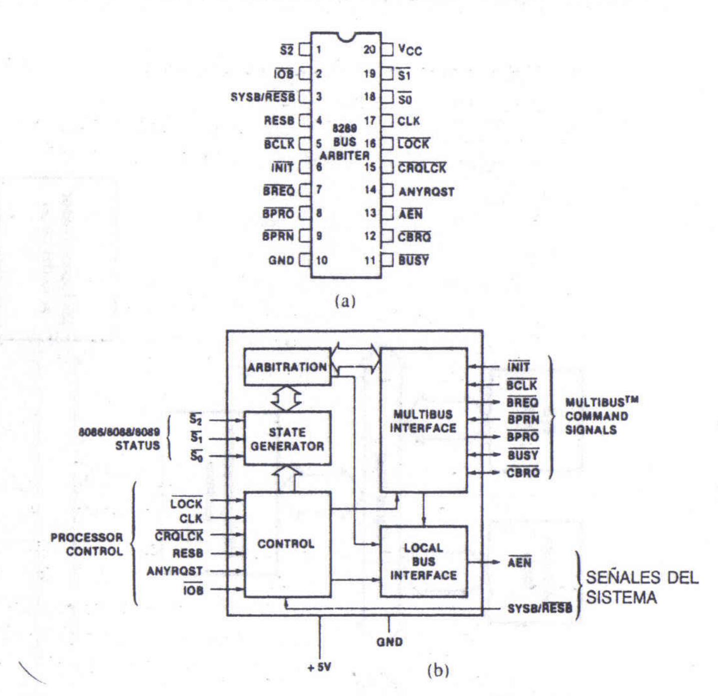
Antes de poder entender a fondo la figura 11-17 hay que captar la operación del árbitro de canales. El árbitro 8289 controla la interface de un amo de canal a un canal compartido. Aunque el 8289 no es el único árbitro de canal, está construido para funcionar con los microprocesadores 8086 y 8088, por lo cual se va a describir aquí. Cada amo de canal microprocesador requiere un árbitro para la interface con el canal compartido, Multibus, que es marca registrada de Intel Corporation y Micro Channel de la IBM. El canal compartido se utiliza sólo para transmitir información de un microprocesador a otro; de lo contrario, los amos de canales funcionan en sus propios modos de canal local y emplean sus propios programas, memoria y espacio de E/S locales. A los microprocesadores conectados en un sistema como éste, a menudo se les llama procesadores *paralelos* o *distribuidos*, porque pueden ejecutar programas y realizar tareas en paralelo.

Arquitectura del 8289

En la figura 11-16 se ilustran los diagramas a bloques y de terminales del árbitro de canal 8289. En el lado izquierdo del diagrama de bloque se ilustran las conexiones con el microprocesador. En el lado derecho se presenta la conexión del 8289 con el canal compartido (remoto) o Multibus.

El 8289 controla el canal compartido porque haciendo que la entrada READY del microprocesador se convierta en 0 lógico (no está listo) si se le niega el acceso al canal compartido. Este *bloqueo* ocurre siempre que otro microprocesador accesa al canal compartido. Como resultado, el microprocesador que solicita acceso queda bloqueado por el 0 lógico aplicado a la entrada READY; cuando esta terminal es un 0 lógico, el microprocesador y su programa esperan hasta que el árbitro les conceda el acceso al canal compartido. En esta forma, sólo un microprocesador a la vez logra acceso al canal compartido. No se requieren instrucciones especiales para el arbitraje de canales con el 8289 porque el arbitraje sólo lo efectúa con circuito.

FIGURA 11-16 Diagrama de base y diagrama a bloques del 8289. (Cortesía de Intel Corporation.)



Definición de terminales

1. $\overline{AEN}$ (salida de habilitación de dirección), hace que los manejadores de canal en un sistema pasen al tercer estado, alta impedancia.
2. ANYRQST (entrada de cualquier solicitud), es una opción de acoplamiento que evita que un microprocesador con menor prioridad pueda ganar acceso al canal compartido. Si se lleva a 1 lógico, ocurre el arbitraje normal y un microprocesador con menor prioridad puede ganar acceso al canal compartido si CBRQ también es un 0 lógico.
3. $\overline{BCLK}$ (entrada del reloj del canal) sincroniza a todos los maestros de canal compartido.
4. $\overline{BPRN}$ (entrada de prioridad al canal) permite al 8289 adquirir el canal compartido en el siguiente flanco de bajada de la señal $\overline{BCLK}$.
5. $\overline{BPRO}$ (salida de prioridad del canal) es una señal que se emplea para decidir la prioridad en un sistema que tiene múltiples amos maestros de canal.
6. $\overline{BREQ}$ (salida de solicitud de canal), se emplea para solicitar acceso al canal compartido.
7. $\overline{BUSY}$ (entrada/salida ocupada) indica, como salida, que el 8289 ha adquirido un canal compartido. Como entrada, se emplea $\overline{BUSY}$ para detectar que otro 8289 ha adquirido el canal compartido.
8. $\overline{CBRQ}$ (entrada/salida solicitud del canal común), se emplea cuando un microprocesador de menor prioridad solicita el uso del canal compartido. Cuando esta señal es una salida se convierte en un 0 lógico siempre que el 8289 solicita el canal compartido y permanece en bajo hasta que el 8289 logra el acceso al canal compartido.

9. CLK (entrada de reloj). La genera el generador de reloj 8284A y constituye la fuente interna de temporización para el 8289.
10. $\overline{CRQLCK}$ (entrada de bloqueo de solicitud común) impide que el 8289 el control de canal a cualquier otro de los 8289 en el sistema. Esta señal funciona en combinación con la terminal $\overline{CBRQ}$.
11. $\overline{INIT}$ (entrada para inicialización) reinicializa al 8289 y se conecta con la señal de RESET del sistema.
12. $\overline{IOB}$ (entrada de E/S) selecciona si el 8289 funciona en un sistema de canal compartido (si se seleccionó por RESB), con E/S ($\overline{IOB}$ = 0) o con la memoria y E/S ($\overline{IOB}$ = 1).
13. $\overline{LOCK}$ (entrada) evita que el 82829 deje que otros microprocesadores tengan acceso al canal compartido. Una instrucción del 8086/8088 que lleve el prefijo LOCK impedirá que otros microprocesadores accesen el canal compartido.
14. RESB (entrada al canal residente) es una conexión de acoplamiento que le permite al 8289 funcionar en un sistema que tiene sistema de canal compartido o de canal residente. Si RESB es un 1 lógico, se configura al 8289 como amo del canal compartido. Si RESB es un 0 lógico, se configura a 8289 como amo del canal local. Cuando esté configurado como amo del canal compartido, el acceso se solicita en la terminal de entrada SYSB/$\overline{RESB}$.
15. $\overline{S2}$, $\overline{S1}$ y $\overline{S0}$ (entradas de estado) inician las solicitudes y cesiones del canal compartido. Estas terminales se conectan con las terminales de estado del controlador de canal 8288.
16. SYSB/$\overline{RESB}$ (entrada canal del sistema/canal residente) selecciona el sistema de canal compartido cuando se le pone en 1 lógico o el canal residente local cuando se le pone en 0 lógico.

Funcionamiento general del 8289. Como se señala en la descripción de las terminales, se puede hacer funcionar al 8289 en tres modos básicos: (1) modo de canal periférico de E/S; (2) modo de canal residente; (3) modo de canal único. En la tabla 11-1 aparecen las conexiones requeridas para que el 8289 funcione en esos modos. En el *modo* de *canal periférico de E/S*, todos los componentes que hay en el canal local se consideran como E/S incluso la memoria, y se accesa a ellos con instrucciones para el espacio de entrada/salida. Todas las referencias a la memoria permiten acceso al canal compartido y todos los dispositivos de E/S acceden al canal local residente. El *modo de canal residente* permite accesos a la memoria y a E/S en los canales local y compartido. Asimismo, el *modo de canal sencillo* sirve de interface del microprocesador con un canal compartido pero el microprocesador no tiene memoria ni E/S locales. En muchos sistemas, el microprocesador se inicializa como el amo de canal compartido (modo de canal sencillo) para controlar y poder ser amo del canal compartido. El *amo del canal compartido* controla al sistema por medio de la memoria y E/S compartidos. Hay microprocesadores adicionales conectados con el canal compartido como amos de canal residente o periféricos de E/S. Estos amos de canal adicionales suelen efectuar tareas independientes, que se reportan al amo del canal compartido a través de éste.

TABLA 11–1 Modos de funcionamiento del 8289

Modo	*Conexiones de terminales*
Single bus	$\overline{IOB}$ = 1 y RESB = 0
Resident bus	$\overline{IOB}$ = 1 y RESB = 1
I/O bus	$\overline{IOB}$ = 0 y RESB = 0
I/O bus y resident bus	$\overline{IOB}$ = 0 y RESB = 1

Sistema para ilustrar las conexiones de canal sencillo y canal residente. En la operación con canal sencillo un microprocesador está en interface con un canal compartido que contiene los recursos de E/S y de memoria que se comparten con otros microprocesadores. En la figura 11-17 se ilustran tres microprocesadores 8088, cada uno conectado con un canal compartido. Dos de estos microprocesadores funcionan en el modo de canal residente; el tercero funciona en el modo de canal sencillo. El microprocesador A (figura 11-17) funciona en el modo de canal sencillo y no tiene canal local. Este microprocesador sólo accesa a memoria y espacio de E/S compartidos. Al microprocesador A se le llama, a menudo, el *amo del canal del sistema* porque es el que se encarga de coordinar las tareas de la memoria principal y de E/S. Los otros dos microprocesadores (B y C) están conectados en el modo de canal residente, lo cual les permite acceso al canal compartido y a sus propios canales locales. Estos microprocesadores de canal residente se emplean para ejecutar tareas independientes del amo de canal del sistema. En realidad, el único momento en que se interrumpe al amo del canal del sistema de realizar sus tareas es cuando uno de los dos microprocesadores con canal residente necesita transferir datos entre él mismo y el canal compartido. Esta conexión permite que los tres microprocesadores ejecuten sus tareas sumultáneamente, pero cuando se necesite, se pueden compartir datos entre los microprocesadores.

En la figura 11-17 el amo (A) de canal permite al usuario trabajar con una terminal de video que permite la ejecución de programas y, en general, controla el sistema. El microprocesador B maneja todas las comunicaciones telefónicas y pasa la información, por bloques, a la memoria compartida. Esto significa que el microprocesador B espera a que se transmita o reciba cada carácter y controla el protocolo utilizado para las transferencias. Por ejemplo, supóngase que se transmite un bloque de datos de 1 Kbyte por medio de la interface telefónica a una velocidad de 100 caracteres por segundo. Esto significa que la transferencia requiere 10 segundos. En vez de inmovilizar al amo de canal durante 10 segundos, el microprocesador B, "pacientemente" realiza la transferencia de datos desde su propia memoria local y la interface local para comunicaciones. Esto libera al amo del canal para otras tareas. La única vez en que el microprocesador B interrumpe al amo del canal, es para transferir datos entre la memoria compartida y el sistema de memoria local. Esto sólo requiere unos cuantos cientos de microsegundos del amo del canal y del sistema principal.

El microprocesador C se utiliza para formar un área de impresión. Su única tarea es imprimir datos en la impresora. Siempre que el amo del canal requiere una salida impresa, transfiere la tarea al microprocesador C; éste, entonces, accede a la memoria compartida, captura los datos que se van a imprimir y los almacena en su propia memoria local, desde la cual se imprimen después los datos y se deja libre al amo del canal para ejecutar otras tareas. Esto permite al sistema ejecutar un programa con el amo del canal, transferir datos por la interface de comunicaciones con el microprocesador B e imprimir información en la impresora con el microprocesador C. Todas estas tareas se ejecutan al mismo tiempo. No hay límite del número de microprocesadores conectados en un sistema ni en el número de tareas que se efectúen al mismo tiempo con esta técnica. El único límite es el producido por el diseño del sistema y por la habilidad del ingeniero.

Lógica de prioridad empleando el 8289

En las aplicaciones en que se emplea el 8289, siempre hay más de un microprocesador conectado con un canal compartido. Debido a que sólo puede haber un acceso cada vez al canal compartido, se debe emplear algún método para establecer la prioridad. La prioridad evita que más de un microprocesador accese al canal en un momento dado. Hay dos métodos para establecer la prio-

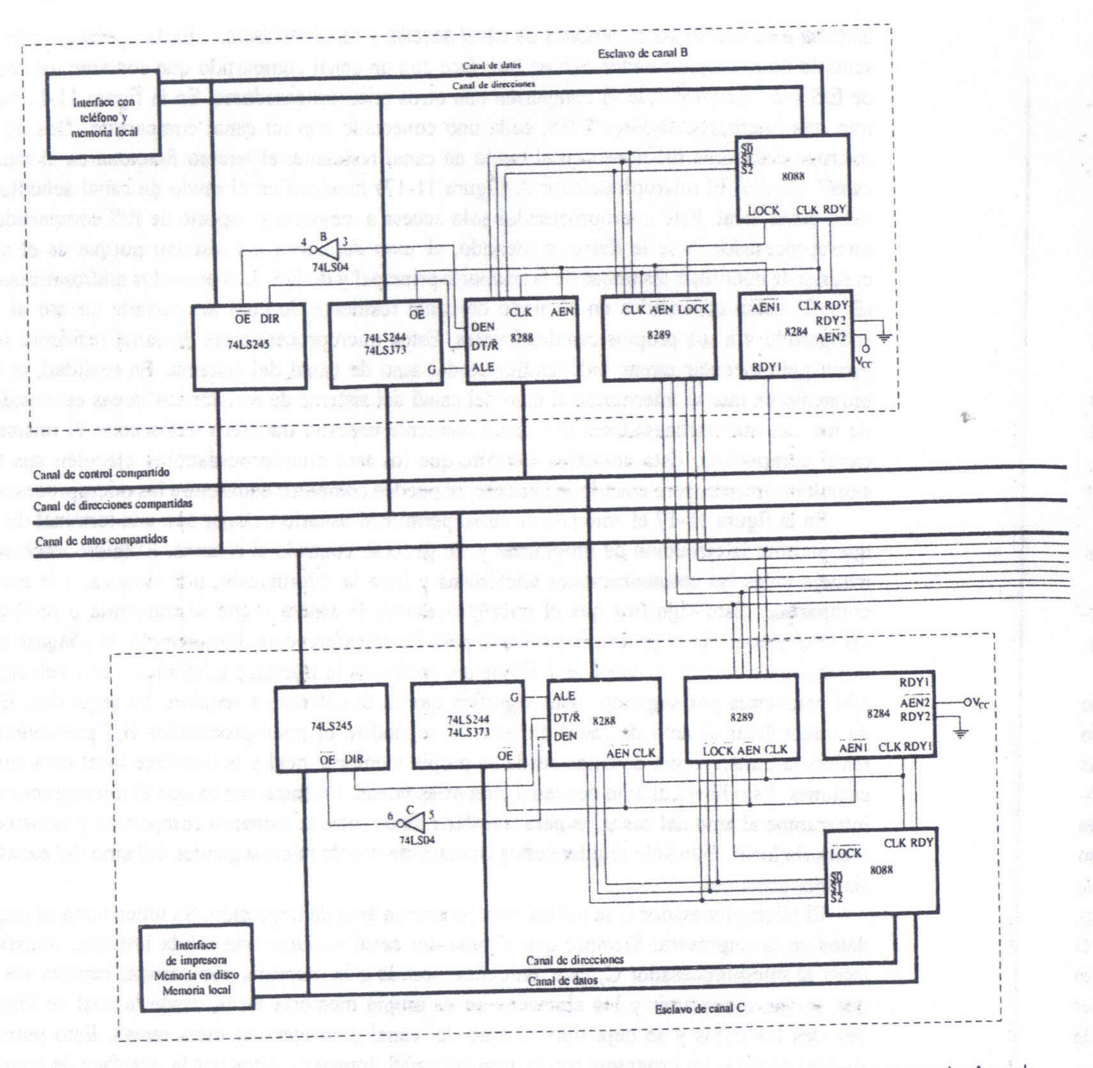


FIGURA 11–17 Tres microprocesadores 8088 que comparten un canal de sistema común. El microprocesador A es el amo del canal que controla la memoria compartida y la terminal CRT. El microprocesador B es esclavo y controla la interface y la memoria telefónica locales. El microprocesador C es también un esclavo que controla una impresora, el sistema de memoria en disco y la memoria local.

ridad con el árbitro de canal 8289: los métodos de cadena de margaritas (serial) y de prioridad paralelas.

Prioridad en cadena de margaritas. Con este método se conecta la salida $\overline{\text{BPRO}}$ con la entrada $\overline{\text{BPRN}}$ al 8289 que tenga la prioridad inmediata inferior; es el menos costoso de implantar. En la figura 11-18 se ilustra el esquema de cadena de margaritas para conectar cierto número de 8289

en un sistema. Debido a que la entrada $\overline{\text{BPRN}}$ del 8289 con máxima prioridad está conectada a tierra, recibe una aceptación inmediata siempre que el microprocesador solicita acceso al canal compartido. La salida $\overline{\text{BPRO}}$ es un 0 lógico si el 8289 y el microprocesador están activos y un 1 lógico si éstos utilizan en forma activa el canal compartido. Si no hay solicitudes activas, todas las entradas $\overline{\text{BPRN}}$ detectarán un 0 lógico. Tan pronto como el 8289 de máxima prioridad recibe

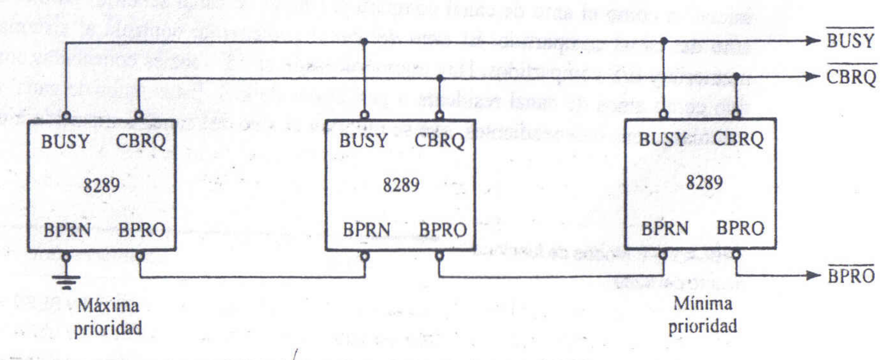


FIGURA 11–18 Resolvedor de prioridad, en cadena de margaritas, en el 8289.

un reconocimiento del canal, su salida $\overline{\text{BPRO}}$ va a alta y bloquea el acceso de todos los 8289 de menor prioridad al canal compartido. Si más de un 8289 recibe reconocimiento, entonces más de uno funcionará al mismo tiempo. Este método rara vez se emplea porque dos microprocesadores pueden acceder, al mismo tiempo, al canal compartido. Por esta razón, Intel recomienda que este método se limite a no más de tres 8289 en un sistema en que se use un reloj de canal de 10 MHz o menos. Con un reloj de esta frecuencia, no ocurren problemas y sólo un microprocesador accede, cada vez, al canal compartido. Si se conectan más árbitros, Intel sugiere resolver la prioridad con el método paralelo.

Prioridad paralela

En la figura 11-19 se ilustra un método de prioridad paralela, con cuatro árbitros de canal 8289 conectados con un circuito en paralelo. En este caso, se utiliza un codificador 74LS148 de prioridad de ocho entradas para resolver en paralelo los conflictos de prioridad. En este ejemplo, sólo se emplean cuatro entradas para resolver la prioridad para los cuatro 8289. Las cuatro entradas al codificador no utilizadas se llevan a un nivel positivo 1 lógico para deshabilitar estas entradas no usadas del codificador. Se debe tener en cuenta que este circuito se puede ampliar a fin de dar prioridad a hasta ocho 8289 con sus microprocesadores.

El circuito de la figura 11-19 funciona como sigue: si todos los 8289 están inactivos (no hay solicitudes al canal compartido en las entradas SYSB/$\overline{\text{RESB}}$ todas las salidas $\overline{\text{BREQ}}$ están en alto y las salidas del 74LS148 son 1 lógico (A y B son 1). Esto significa que el 8289 con máxima prioridad accederá al canal compartido si lo solicita su microprocesador. Por otra parte, si se hace una solicitud de menor prioridad, la salida $\overline{\text{BREQ}}$ se convierte en 0 lógico. Esto hace que el codificador de prioridad ponga un 0 lógico en la terminal de entrada $\overline{\text{BPRN}}$ del 8289 correspondiente para permitir acceso al canal compartido. Por ejemplo, si el 8289 que está a la extrema derecha pone un 0 lógico en su terminal de salida $\overline{\text{BREQ}}$, el codificador tendrá un cero en su entrada $\overline{3}$. Esto hace que el 74LS148 genere un 00 en sus terminales de salida. Este 00 hace que el 74LS138 active la entrada BPRN en el 8289 de la extrema derecha y le da acceso al canal compartido. También bloquea cualquier otra solicitud porque la señal $\overline{\text{BUSY}}$ se vuelve un 0 lógico. Si ocurren solicitudes simultáneas, el 74L5148 le otorga la prioridad en forma automática, para evitar conflictos sin que importe cuántos 8289 están conectados en un sistema. Por esta razón, es deseable este esquema de la priorización («prioridad»).

Cola de impresión e interface para imprimir

En la figura 11-20 se presenta el diagrama de bloque de una cola de impresión («spooler») y una interface para impresora, controladas por un microprocesador 8088. En este caso, hay dos microprocesadores en el sistema; uno, que es el amo del canal del sistema, funciona en el modo de canal sencillo y el segundo funciona en el modo de canal residente. Debido a que hay dos microprocesadores, uno puede imprimir los datos en tanto el otro se emplea para procesar nueva información.

En esta interface, el microprocesador esclavo transfiere datos a la impresora desde su memoria local, sin intervención del amo de canal del sistema. Los datos se transfieren a la memoria local del microprocesador esclavo siempre que accesa a la memoria compartida para pedir datos adicionales.

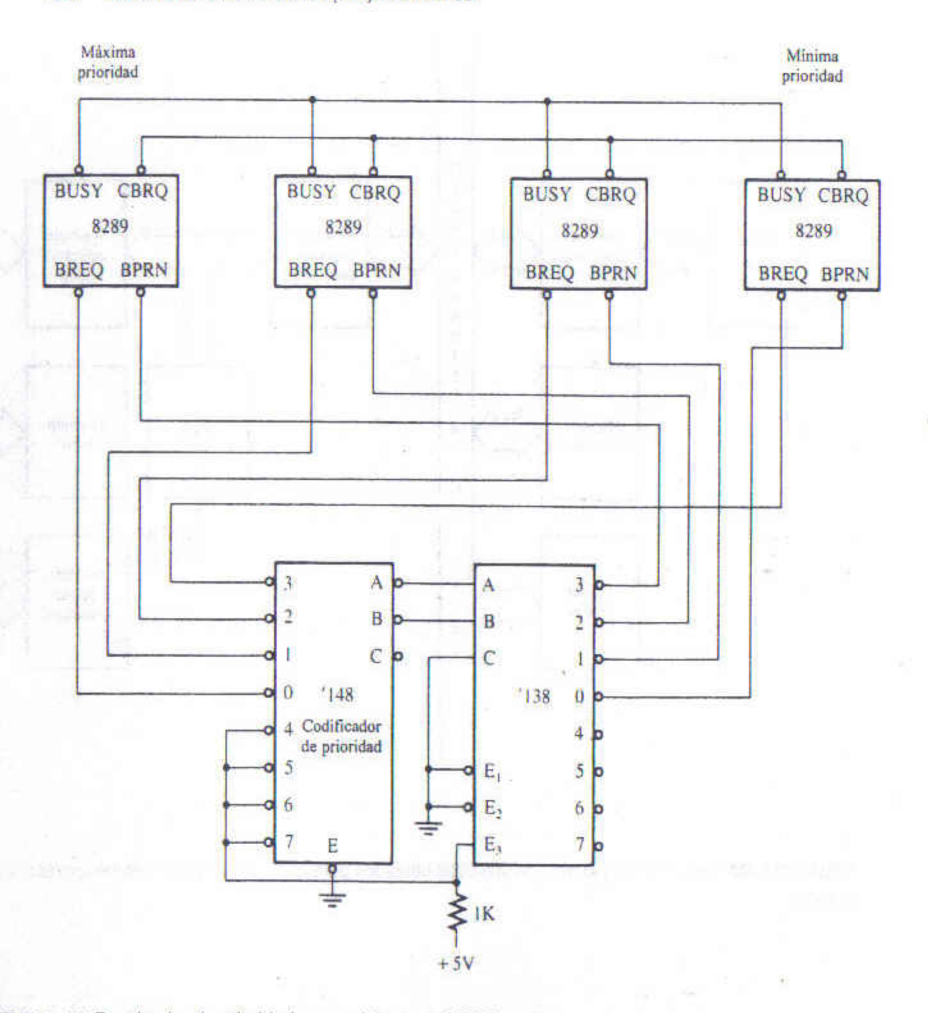


FIGURA 11-19 Resolvedor de prioridad en paralelo para el 8289.

Interface maestro-canal en el modo sencillo. En la figura 11-21 se ilustra el 8088 amo del canal que funciona en el modo sencillo (en interface) con el canal compartido. El amo del canal compartido tiene acceso a todas las localidades de memoria y dispositivos de E/S en ese canal. La señal $\overline{BCLK}$ la genera un generador de reloj 8284A que se utiliza como oscilador de 10 MHz.

El árbitro 8289 funciona sin ningún canal de E/S, ni canal residente y con la terminal ANYRQST conectada a nivel alto para que se pueda acceder al canal compartido para todas las transferencias de memoria y de E/S. En este sistema se emplea el método de prioridad de cadena de margaritas, lo cual significa que la señal $\overline{BREQ}$ no está conectada. La interface permite que el microprocesador esclavo acceda a la memoria compartida siempre que sea necesario y, con ello, evita que el microprocesador esclavo cierre el canal durante demasiado tiempo.

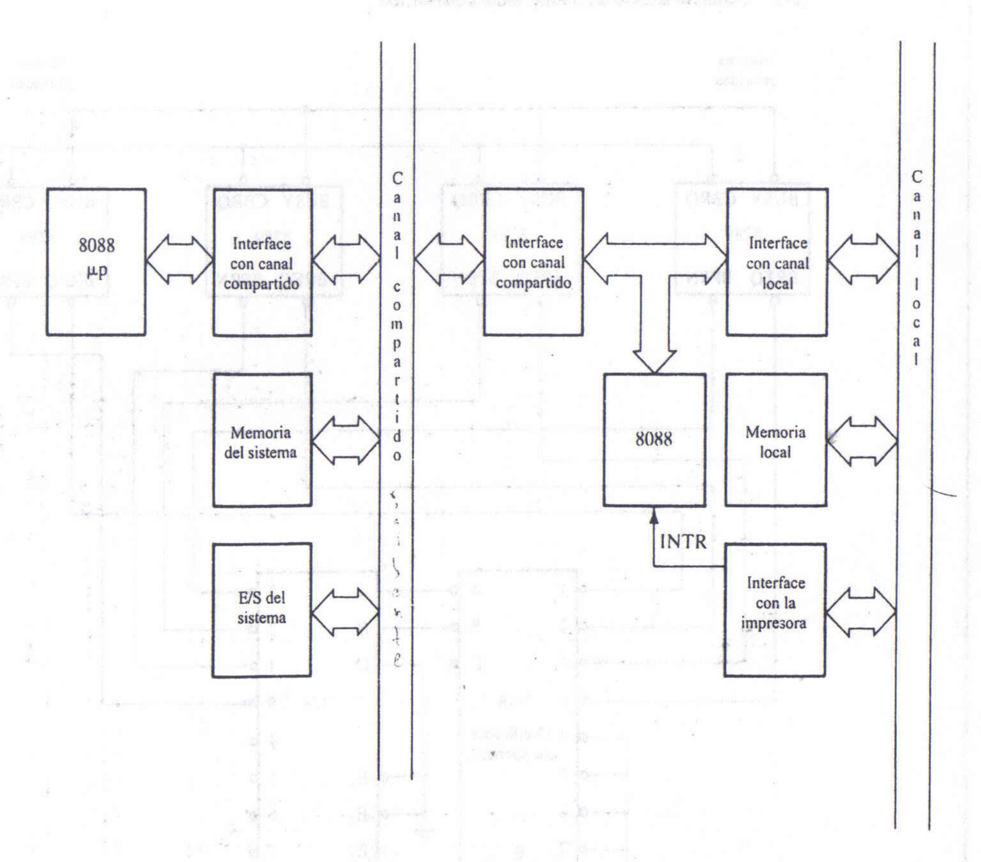


FIGURA 11-20 Diagrama de bloques de dos 8088 utilizados para controlar una interface de impresora y una cola de impresión.

Además de las señales normales del canal, este circuito también proporciona al canal compartido una señal PCLK de 2.5 MHz para cualquier dispositivo de E/S. Se ilustra la entrada RDY en caso de que la memoria o los dispositivos de E/S del sistema requieran estados de espera.

Funcionamiento del 8088 esclavo con canal residente. En la figura 11-22 se ilustra el microprocesador esclavo 8088 que forma la cola de impresión. El microprocesador está conectado como amo del canal residente. Se ilustran la interface entre el canal local y el residente y la interface con el canal compartido.

Siempre que el 8088 esclavo accede a la memoria más arriba de la localidad 7FFFFH, coloca en 1 lógico la entrada SYS/$\overline{\text{RESB}}$ del 8289 y la terminal CEN del amo de canal compartido 8288. Así solicita acceso al canal compartido a través del árbitro 8289. Si la dirección está por abajo de 80000H, se pone a tierra la terminal, $\overline{\text{SYS}}$/$\overline{\text{RESB}}$ del 8289 y se coloca un 1 lógico en la terminal CEN del 8288 del canal local. Con ello, se solicita acceso al canal local residente para el control de la impresora y de la memoria del canal local residente. Se debe tener en cuenta que no se

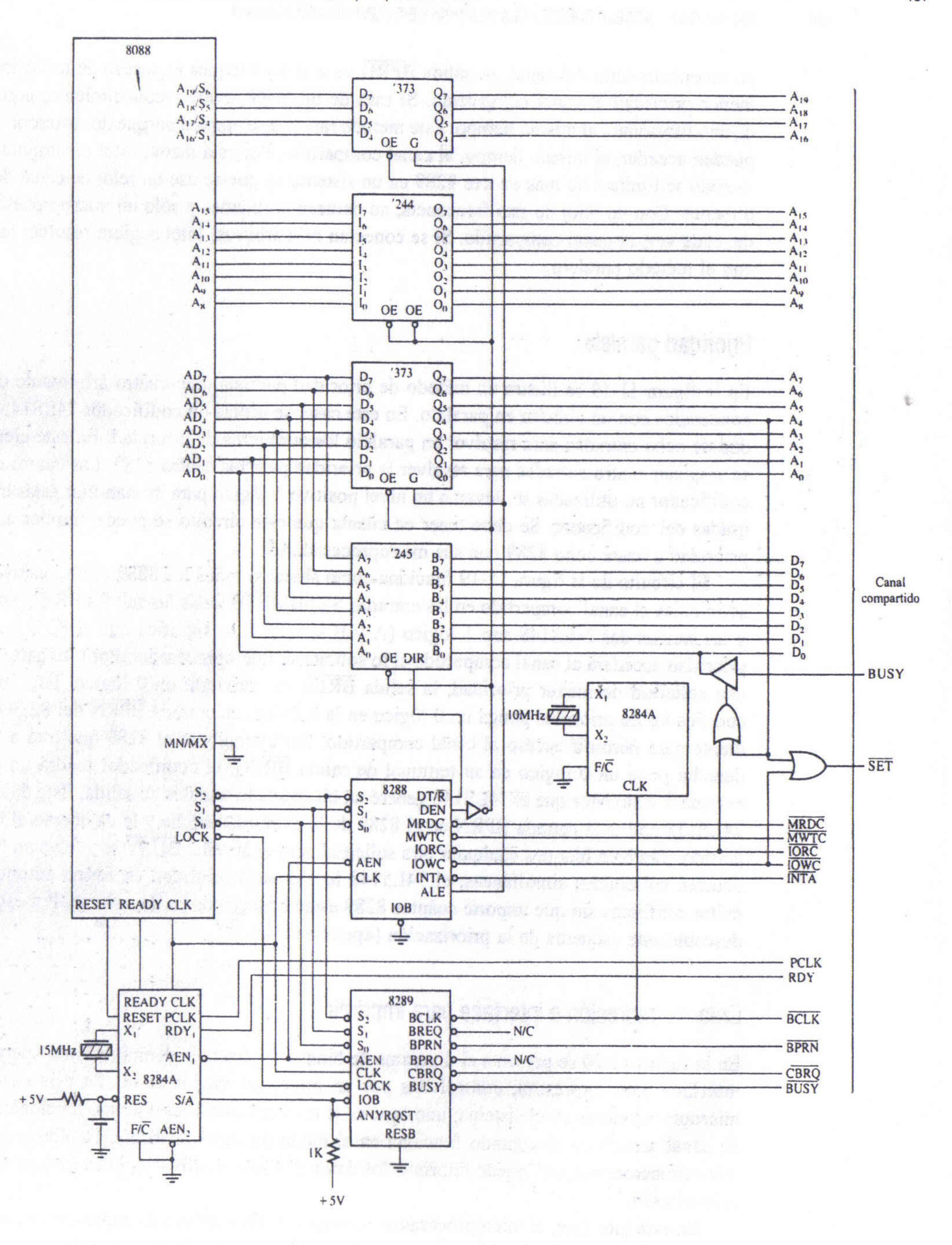


FIGURA 11-21 Un 8088 conectado para el modo de funcionamiento de canal sencillo.

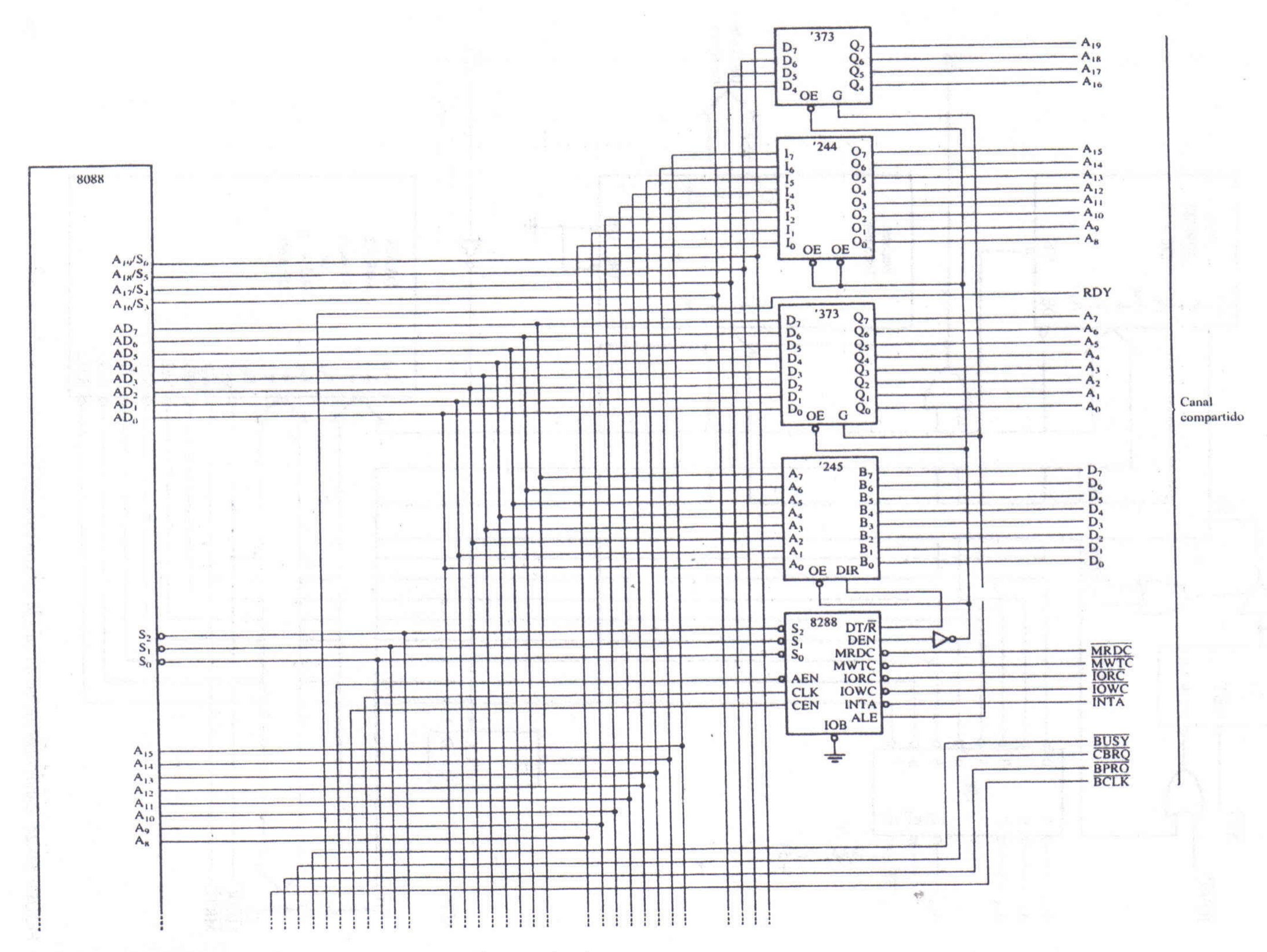


FIGURA 11–22 Se ilustra el 8088 con un canal compartido y uno local.

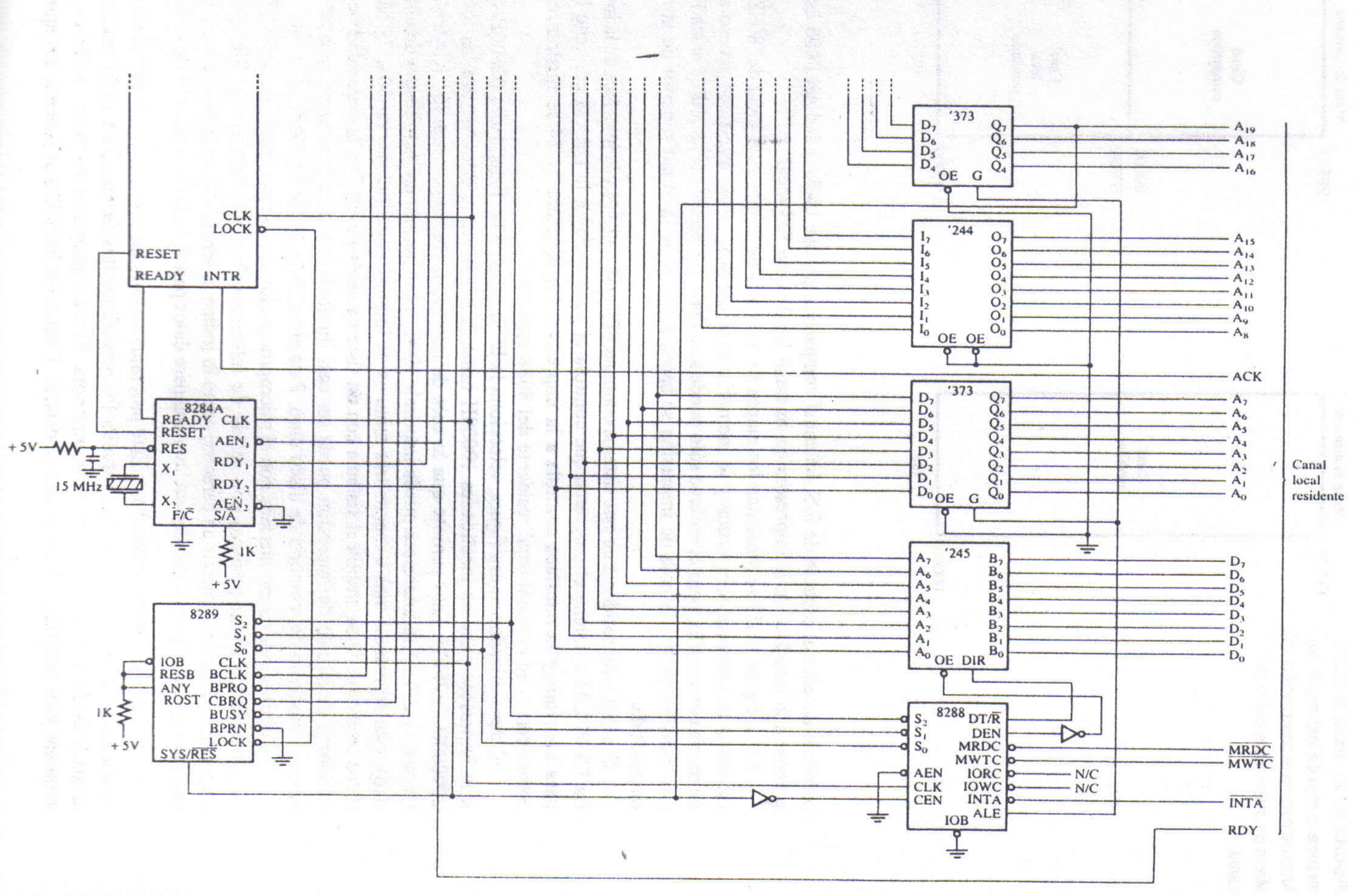
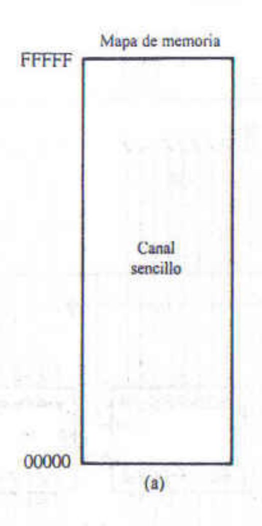


FIGURA 11–22 (Continúa)



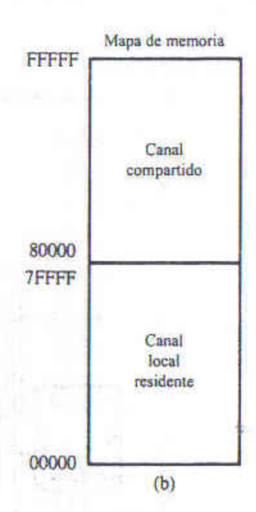


FIGURA 11-23 Mapa de memoria para la cola de impresión. (a) Amo de memoria compartida. (b) Mapa de memoria esclavo del canal.

pretende acceder al espacio de E/S del canal compartido porque la finalidad del 8088 esclavo es acceder a la interface de la impresora en su canal local como E/S local.

En la figura 11-23 se presentan los mapas de memoria de la memoria local del 8088 esclavo así como la memoria compartida y la memoria local del 8088 amo; éste tiene acceso a toda la memoria compartida, pero el esclavo sólo puede acceder a las localidades 80000H hasta FFFFFH. Las transferencias a la cola de impresión se hacen a través de la mitad superior de la memoria compartida.

El canal local residente en este sistema contiene EPROM, DRAM y la interface de la impresora. La EWPROM contiene el programa que controla al 8088 esclavo; la DRAM contiene los datos para imprimir y la interface controla a la impresora. En la figura 11-24 se ilustran estos tres componentes en el canal local residente del 8088 esclavo.

El decodificador en esa figura, selecciona la EPROM para las localidades 00000H-0FFFFH, a la impresora para las localidades 10000H hasta 7FFFFH y a la DRAM para las localidades 200000H-7FFFFH. Esto permite que la cola de impresión contenga hasta 364 Kbytes de datos (figura 11-25). Si la impresora puede imprimir 1000 líneas por minuto (impresora de alta velocidad), esto representa más o menos tres minutos de impresión y alrededor de 50 a 75 páginas de datos impresos. Si se amplía el sistema con un sistema de memoria fija, la capacidad de almacenamiento de la cola de impresión puede ser casi ilimitada debido a la enorme capacidad de un sistema moderno de memoria de disco duro. Para simplificar, no se ha añadido en este ejemplo.

La impresora está en interface con el microprocesador esclavo por medio de un 74LS244 que funciona como un dispositivo con salida de interrrupción (estroboscópica). Siempre que la impresora acaba de imprimir un caracter, como lo indica la señal $\overline{ACK}$ se genera una interrupción que hace que el microprocesador esclavo recupere datos para la cola de impresión de la impresora.

Programa para la cola de impresión. Este programa es muy directo. El listado completo se suministra en secciones, a fin de poder estudiar los procedimientos de inicialización, transferencia de datos, y del servicio de interrupción la impresora. El único programa que no se ilustra es el que se requiere para programar e inicializar el sistema. Cuando se inicializa el sistema, los apuntadores

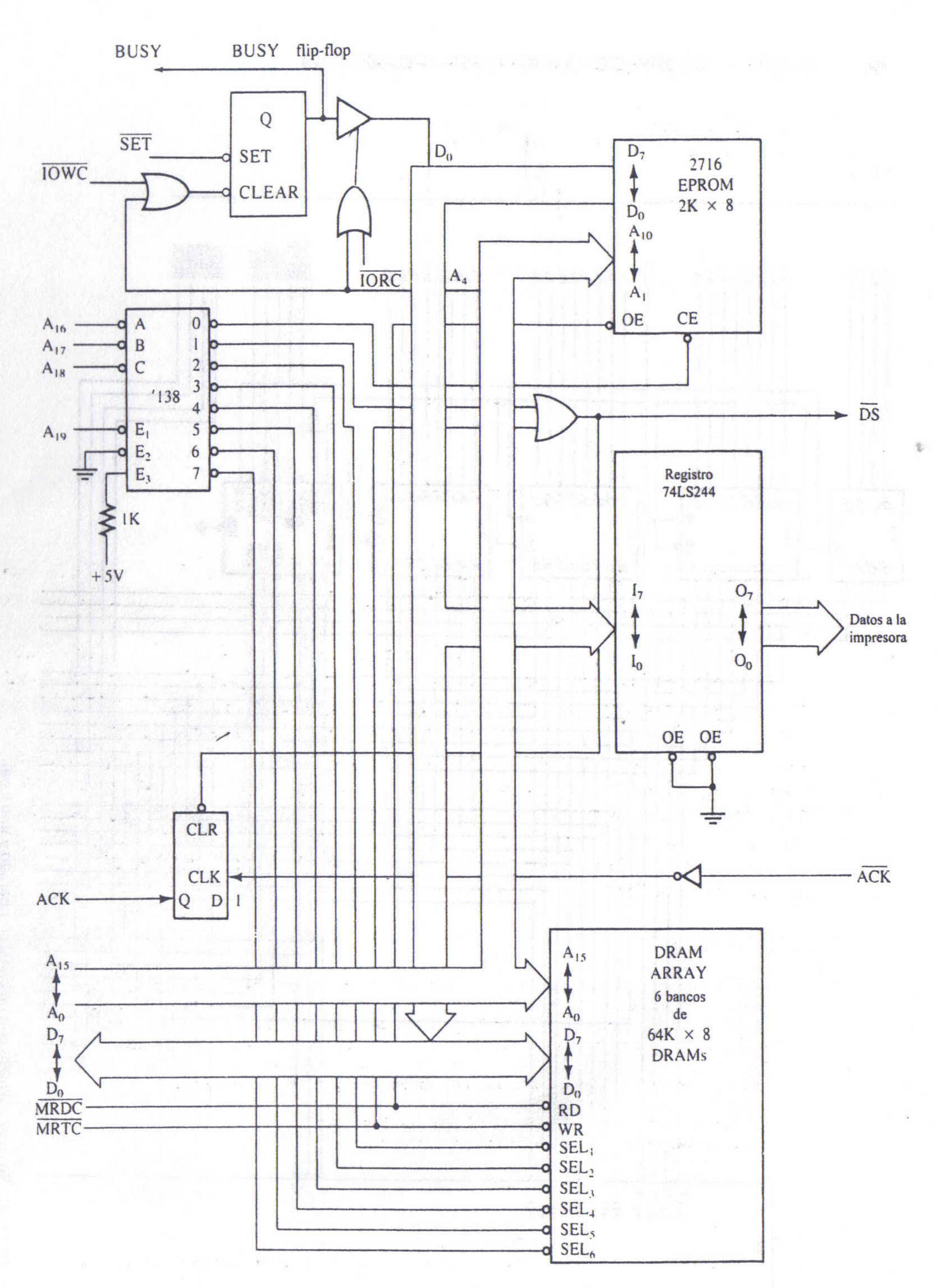
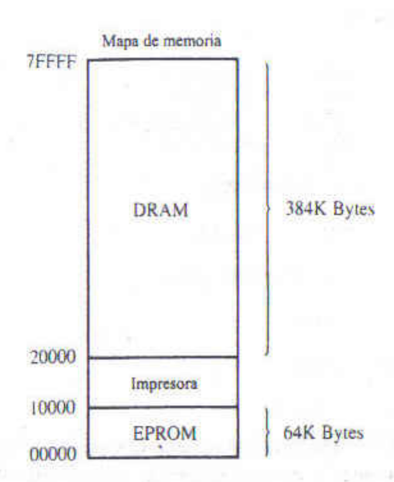
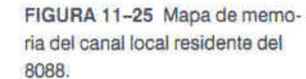


FIGURA 11–24 Canal local residente para la cola de impresión.




FIGURA 11–25 Mapa de memoria del canal local residente del 8088.

de entrada y salida de la cola están inicializados con la dirección 20000H. Esta situación (apuntadores iguales) indica que la cola está vacía. El segmento de los apuntadores contiene a 2000H y el desplazamiento contiene a 0000H.

En el ejemplo 11-5 se lista el programa que transfiere datos a la cola de impresión, desde el canal compartido. En este programa, el amo del canal carga un bloque de la memoria compartida con los datos para la impresora, llamada BUFFER (que empieza en la localidad 80002H) y luego manda una señal mediante un flip-flop al 8088 esclavo de que hay datos disponibles, lo cual sirve de indicador para el esclavo. El flip-flop de bandera el amo lo activa siempre que hay datos disponibles para la impresora y lo desactiva el esclavo después que ha retirado los datos de la memoria compartida. También se coloca información adicional en la memoria compartida para el esclavo: la longitud del bloque de datos. La localidad 80000H contiene una palabra que indica la longitud del bloque de los datos para impresión. El tamaño máximo del área de la memoria BUFFER es de 64 Kbytes.

EJEMPLO 11–5

```
                ;Procedimiento que hay en el microprocesador amo que prueba la bandera (FLAG)
                ;para determinar si el esclavo está ocupado. Si el esclavo no está ocupado,
                ;se transfieren los datos hacia la memoria de impresión y se activa la bandera
                ;para indicar que hay disponibles datos para que el esclavo los imprima.
                ;
                ;El contenido de CX = el número de caracteres que se van a cargar

                ;El contenido de DS:SI = la localidad de los datos
                ;que se van a enviar al esclavo para imprimir.
                ;
0000                        LOAD    PROC  FAR

0000 E4 00                          IN    AL,00H
0002 A8 01                          TEST  AL,1
0004 75 FA                          JNZ   LOAD      ;esperar a que termine el esclavo

0006 06                             PUSH  ES
0007 BF 0002 R                      MOV   DI,OFFSET BUFFER
```

```
000A B8 8000              MOV   AX,8000H
000D 8E C0                MOV   ES,AX
000F 26: 89 0E 0000 R     MOV   ES:LENGTH,CX

0014 F3/ A4               REP   MOVSB

0016 07                   POP   ES
0017 B0 FF                MOV   AL,0FFH
0019 E6 00                OUT   00H,AL    ;activar bandera de datos disponibles

001B CB                   RET

001C            LOAD      ENDP
```

Una vez que el esclavo detecta un 1 lógico en el flip-flop de la bandera, empieza a transferir datos desde la memoria compartida a su propia memoria local. Los datos se almacenan en la memoria local que está organizada como FIFO (primero en entrar, primero en salir) o cola. En este ejemplo, la longitud de la cola de impresión es de 384 Kbytes. En el ejemplo 11-6 se presenta el programa que utiliza el esclavo para cargar la cola de impresión desde el área de la memoria compartida. En este programa, el esclavo prueba el flip-flop de bandera para determinar si el amo ya llenó el área de impresión en la memoria compartida. Si el área de impresión está llena, el microprocesador esclavo transfiere el contenido de la memoria compartida hacia la cola de impresión. Una vez concluida la transferencia, el esclavo desactiva el flip-flop de bandera para que el amo pueda empezar a llenar el área de impresión de la memoria con datos adicionales para la impresora.

EJEMPLO 11-6 (página 1 de 3)

```
                ;Procedimiento que hay en el microprocesador esclavo y que prueba la
                ;bandera FLAG para determinar si hay datos disponibles para imprimir.
                ;Si los hay, este procedimiento los transfiere a la memoria local
                ;del procesador esclavo para imprimir.
                ;
0000                  TRANSFER  PROC   FAR

0000 E4 00                      IN     AL,00H
0002 A8 01                      TEST   AL,1
0004 74 FA                      JZ     TRANSFER       ;si no hay datos

0006 B8 8000                    MOV    AX,8000H       ;cargar segmentos
0009 8E D8                      MOV    DS,AX
000B BE 0000                    MOV    SI,0           ;cargar fuente
000E 8B 0C                      MOV    CX,[SI]        ;obtener conteo
0010 83 C6 02                   ADD    SI,2           ;ajustar apuntador

0013 E8 0023 R                  CALL   FILL_QUEUE     ;llenar cola de instrucciones

0016 FB                         STI                   ;habilitar interrupción

0017 E6 00                      OUT    00H,AL         ;borrar flip-flop
0019 EB E5                      JMP    TRANSFER

001B                  TRANSFER  ENDP

001B 0000             IN_POINT_SEG    DW    ?         ;apuntador de entrada
001D 0000             IN_POINT_OFF    DW    ?
```

EJEMPLO 11–6 (página 2 de 3)

```
001F 0000           OUT_POINT_SEG      DW      ?          ;apuntador de salida
0021 0000           OUT_POINT_OFF      DW      ?

          ;Procedimiento que transferirá datos desde la memoria del amo
          ;a la cola de impresión en el esclavo. DS:SI direccionan los datos
          ;en la memoria compartida y CX contiene el contador.
          ;
0023                FILL_QUEUE PROC    NEAR

0023 FC                        CLD
0024 2E: A1 001B R             MOV     AX,CS:IN_POINT_SEG
0028 8E C0                     MOV     ES,AX
002A 2E: 8B 3E 001D R          MOV     DI,CS:IN_POINT_OFF

002F                           FULL:

002F E8 0046 R                 CALL    TEST_FULL          ;probar si está llena
0032 74 FB                     JZ FULL                    ;si es que está llena,

0034 A4                        MOVSB
0035 E8 005C R                 CALL    INC_IN_POINT       ;incrementar
                                                          ;apuntador de entrada
0038 8C C0                     MOV     AX,ES
003A 2E: A3 001B R             MOV     CS:IN_POINT_SEG,AX
003E 2E: 89 3E 001D R          MOV     CS:IN_POINT_OFF,DI
0043 E2 EA                     LOOP    FULL            ;repetir para todos los datos
0045 C3                        RET

0046                   FILL_QUEUE      ENDP

          ;Procedimiento para probar una cola completa
          ;
0046                 TEST_FULL  PROC   NEAR

0046 1E                        PUSH    DS
0047 57                        PUSH    DI
0048 E8 005C R                 CALL    INC_IN_POINT
004B 8C D8                     MOV     AX,DS
004D 2E: 3B 06 001F R          CMP     AX,CS:OUT_POINT_SEG
0052 75 05                     JNE     TEST_FULL_END
0054 2E: 3B 3E 0021 R          CMP     DI,OUT_POINT_OFF

0059                           TEST_FULL_END:

0059 5F                        POP     DI
005A 1F                        POP     DS
005B C3                        RET

005C                   TEST_FULL       ENDP

                       ;Procedimiento de incrementar apuntador de entrada
                       ;

005C              INC_IN_POINT PROC    NEAR

005C 47                        INC     DI
005D 0B FF                     OR DI,DI
005F 75 0F                     JNE     INC_IN_POINT_END
0061 8C C0                     MOV     AX,ES
```

EJEMPLO 11-6 (página 3 de 3)

```
0063 05 1000                ADD     AX,1000H
0066 3D 8000                CMP     AX,8000H
0069 75 03                  JNE     INC_IN_POINT1
006B B8 2000                MOV     AX,2000H

006E                        INC_IN_POINT1:

006E 8E C0                  MOV     ES,AX

0070                        INC_IN_POINT_END:

0070 C3                       RET

0071         INC_IN_POINT     ENDP
```

En el ejemplo 11-7 aparece el programa que imprime los datos de la cola. Este programa se ejecuta por interrupciones y, en consecuencia, corre como un programa de fondo casi oculto del programa listado en el ejemplo 11-6, excepto por la instrucción para habilitar la interrupción. Siempre que la interface de la impresora, por medio del flip-flop indica que está lista para aceptar datos adicionales, ocurre una interrupción que llama a este procedimiento de servicio de interrupción. El procedimiento extrae datos de la cola y los envía a la impresora.

EJEMPLO 11-7

```
;Procedimiento de servicio de interrupción que imprime datos de la cola.
;Si se imprimen todos los datos, se deshabilitan las interrupciones
;con este procedimiento.
;
0071                PRINT    PROC    FAR

0071 50                      PUSH    AX          ;salvar registros
0072 55                      PUSH    BP
0073 1E                      PUSH    DS
0074 57                      PUSH    DI
0075 06                      PUSH    ES

0076 E8 009D R               CALL    TEST_EMPTY
0079 75 0E                   JNE     PRINT1      ;si no está vacía

007B 8B EC                   MOV     BP,SP
007D 8B 46 0C                MOV     AX,[BP+12]
0080 80 E4 FD                AND     AH,0FDH     ;desactivar interrupción
0083 89 46 0C                MOV     [BP+12],AX
0086 EB 0F 90                JMP     PRINT_END

0089                PRINT1:

0089 B8 1000                 MOV     AX,1000H
008C 8E C0                   MOV     ES,AX
008E 8A 05                   MOV     AL,[DI]
0090 26: A2 0000 R           MOV     ES:DATA,AL  ;imprimir carácter

0094 E8 00B5 R               CALL    INC_OUT

0097                PRINT_END:
```

```
0097 07                     POP     ES              ;restaurar registros
0098 5F                     POP     DI
0099 1F                     POP     DS
009A 5D                     POP     BP
009B 58                     POP     AX
009C CF                     IRET

009D                PRINT   ENDP

009D                TEST_EMPTY      PROC    NEAR

009D 2E: A1 001F R          MOV     AX,CS:OUT_POINT_SEG
00A1 8E D8                  MOV     DS,AX
00A3 2E: 8B 3E 0021 R       MOV     DI,CS:OUT_POINT_OFF
00A8 2E: 3B 06 001B R       CMP     AX,CS:IN_POINT_SEG
00AD 75 05                  JNE     TEST_EMPTY_END
00AF 2E: 3B 3E 001D R       CMP     DI,CS:IN_POINT_OFF
00B4                        TEST_EMPTY_END:

00B4 C3                     RET

00B5                TEST_EMPTY      ENDP

00B5                INC_OUT PROC    NEAR

00B5 47                     INC     DI
00B6 0B FF                  OR      DI,DI
00B8 75 0D                  JNE     INC_OUT_END
00BA 8C C0                  MOV     AX,DS
00BC 05 1000                ADD     AX,1000H
00BF 3D 8000                CMP     AX,8000H
00C2 75 03                  JNE     INC_OUT_END
00C4 B8 2000                MOV     AX,2000H

00C7                        INC_OUT_END:

00C7 2E: A3 001F R          MOV     CS:OUT_POINT_SEG,AX
00CB 2E: 89 3E 0021 R       MOV     CS:OUT_POINT_OFF,DI
00D0 C3                     RET

00D1                INC_OUT ENDP
```

11–4 SISTEMAS DE MEMORIA EN DISCO

La memoria en disco se utiliza para almacenar datos a largo plazo. En la actualidad hay disponibles muchos tipos de sistemas de almacenamiento en discos. En casi todos estos sistemas se utiliza un medio magnético, excepto la memoria de disco óptico en que se almacenan los datos en un disco de plástico. La memoria de disco óptico es un *CDROM* (memoria de sólo lectura para disco compacto), en la cual se lee, pero nunca se escribe o bien una *WORM* (escribir una vez, leer casi siempre), que se lee la mayor parte del tiempo, pero se puede escribir una vez con un rayo laser. También empieza a estar disponible una memoria en disco óptico, en la cual se puede leer y escribir muchas veces, pero todavía hay ciertas limitaciones al número de operaciones de escritura permitidas. En esta sección se presentan los sistemas de memoria en disco para poder emplearlos con sistemas de computadoras; también, se dan detalles del funcionamiento.

Disco flexible

La forma más común y más básica de la memoria en disco es el disco blando, flexible o disquete. Este medio magnético para grabación está disponible en tres tamaños: el *normal* o *estándar* de 8 pulgadas, el *minifloppy* de 5 1/4 pulgadas y el *microfloppy* de 3 1/2 pulgadas. En la actualidad, el tipo de 8 pulgadas ha desaparecido casi por completo y lo han sustituido los discos mini y micro. El disco de 8 pulgadas es en extremo grande y estorboso para manejarlo y almacenarlo. Para resolver este problema, la industria produjo el minidisco de 5 1/4 pulgadas. En la actualidad el microfloppy de 3 1/2 pulgadas va sustituyendo con rapidez al de 5 1/4 en sistemas más modernos, por su tamaño reducido, facilidad de almacenamiento y durabilidad. Aun así, muchos sistemas todavía se venden con ambas unidades para 5 1/4 y 3 1/2 pulgadas. Un fabricante vende una unidad de disco que acepta los de 5 ¼ y 3 1/2 pulgadas.

Todos los discos tienen algunas cosas en común. Todos están organizados de modo de almacenar los datos en **pistas**, la cual es un anillo concéntrico de datos que se almacenan en una superficie de un disco. En la figura 11-26 se ilustra la superficie de un disco de 5 1/4 pulgadas y se muestra una pista dividida en sectores. Un **sector** es una subdivisión común de una pista que se destina a retener una cantidad razonable de datos. En muchos sistemas, un sector contiene 512 o 1024 bytes de datos. El tamaño de un sector puede variar desde 128 bytes hasta toda la longitud de una pista completa.

Se verá en la ilustración que hay una perforación en el disco llamado perforación guía o índice. Esta *perforación guía* está diseñada de modo que el sistema electrónico que lee el disco,

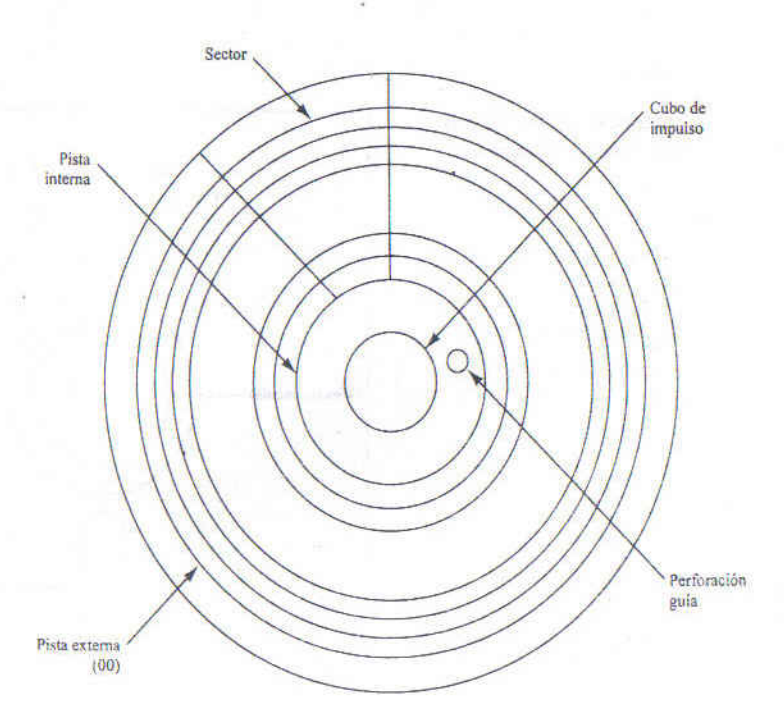


FIGURA 11–26 Formato de un disco blando de 5 1/4 pulgadas.

pueda encontrar el comienzo de una pista y de su primer sector (00). Las pistas están numeradas desde 00, que es la más externa, con números progresivos hacia la pista central o más interna. Los sectores, a menudo, están numerados desde 00 en la pista externa, hasta el valor que se requiera para llegar a la pista interna y a su último sector.

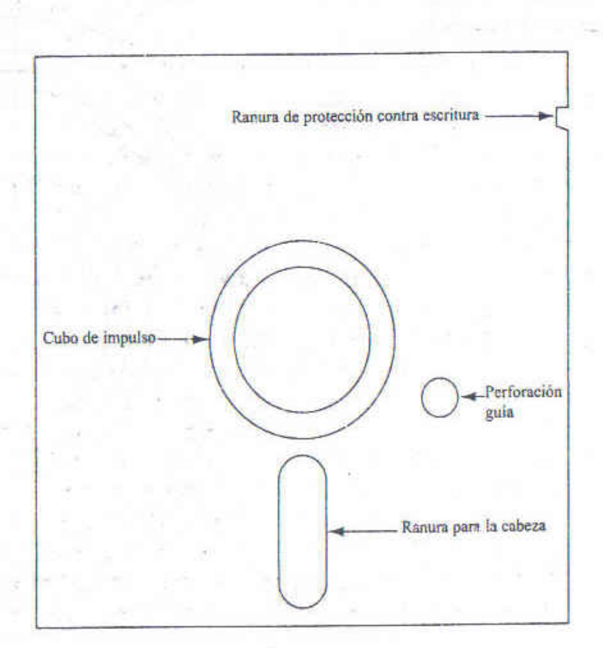
El disco minifloppy de 5 1/4 pulgadas. En la actualidad, el disco de 5 1/4 pulgadas es quizá, el tamaño que más se utiliza en sistemas antiguos de computadora. En la figura 11-27 se ilustra el disquete de 5 1/4 pulgadas, el cual gira a 300 rpm dentro de su funda de plástico semirrígida. El mecanismo de la cabeza en la unidad de disco hace contacto físico con la superficie del disco lo cual, con el tiempo, puede producir desgaste y daños en el disco. Esto podría constituir un problema, pero los discos suelen durar muchos años antes que se pierdan los datos por desgaste.

En la actualidad, casi todos los disquetes son de dos caras o lados. Esto significa que se escriben datos en las superficies superior e inferior del disco. Un conjunto de dos pistas se llama *cilindro* y consta de una pista superior y una inferior. El cilindro 00, por ejemplo, consta de las pistas superior e inferior más externas.

Los datos se almacenan en los disquetes en el formato de doble densidad en el cual se emplea una técnica de grabación que se denomina *MFM* (modulación modificada de frecuencia), para almacenar la información. Los discos de dos lados y doble densidad (*DSDD*) suelen estar organizados con 40 pistas de datos en cada lado del disco. Una pista típica de un disco de doble densidad está dividida en 9 sectores, cada uno con 512 bytes de información. Esto significa que la capacidad total de un disco de dos lados y doble densidad, es de 40 pistas por lado × 2 lados × 9 sectores por pista × 512 bytes por sector o 368 640 (360K) bytes de información.

En los primeros sistemas de memoria en disco se empleaban densidad sencilla y *FM* (modulación de frecuencia) para almacenar la información en 40 pistas en uno o los dos lados del disco. En

FIGURA 11–27 El minidisco blando de 5 1/4 pulgadas.



cada uno de los 8 o 9 sectores del disco de densidad sencilla se almacenaban 256 bytes de datos. Esto significaba que en un disco de densidad sencilla se almacenaban 90K bytes de datos por lado. En un disco de densidad sencilla, de dos lados o caras, se almacenaban 180K bytes de datos.

En la actualidad también son comunes los discos minifloppy de *alta densidad* (HD), que contienen 80 pistas de información por cada lado y 8 sectores por pista. Cada sector contiene 1024 bytes de información. Con esto, el disco de 5 1/4 pulgadas, de alta densidad tiene una capacidad total de 80 pistas por lado, × 2 lados × 8 sectores por pista × 1024 bytes por sector o sea 1,310,720 (1.2 M) bytes de información.

La técnica magnética para grabación que se utiliza para almacenar datos en la superficie de un disco se llama grabación de *no retorno a cero* (NRZ). Al grabar con NRZ, el flujo magnético aplicado en la superficie del disco nunca retorna a cero. En la figura 11-28 se ilustra la información almacenada en una sección de una pista. También se ilustra la forma en que el campo magnético codifica los datos. Se debe tener en cuenta que se utilizan flechas en esta ilustración para mostrar la polaridad del campo magnético almacenado en la superficie del disco.

La razón principal por la que se eligió esta forma de codificación magnética es que borra en forma automática la información vieja cuando se graba nueva información. Si se emplease otra técnica, se requeriría una cabeza borradora separada. La alineación mecánica de una cabeza borradora y de una cabeza separada de lectura y escritura es casi imposible. La densidad del flujo magnético de la señal NRZ es tan intensa que magnetiza (satura) la superficie del disco y borra todo lo que había grabado. También da la certeza de que la información no se alterará con el ruido magnético porque la amplitud del campo magnético no contiene información. La información se almacena en la localidad de los cambios en el campo magnético.

En los sistemas modernos de disco flexible, los datos se almacenan en la forma de modulación de frecuencia modificada (MFM). La técnica MFM para grabación almacena los datos en la forma que se ilustra en la figura 11-29. Se verá que cada tiempo de bit es de 2 microsegundos de ancho en un disco de doble densidad. Cada tiempo de bit de 2 μs se divide en dos partes. Una, retiene un pulso de reloj y la otra retiene un pulso de datos. Si está presente un pulso de reloj, su ancho es de 1 μs igual que la de un pulso de datos. Los pulsos de reloj y datos nunca están presentes al mismo tiempo en un periodo de un bit. (Se debe tener en cuenta que las unidades de los discos de alta densidad reducen estos tiempos a la mitad, por lo cual un tiempo de bit es de 1 μs y un pulso de reloj o de datos tiene 0.5 μs de ancho. Esto duplica la velocidad de transferencia a 1 millón de bits por segundo.)

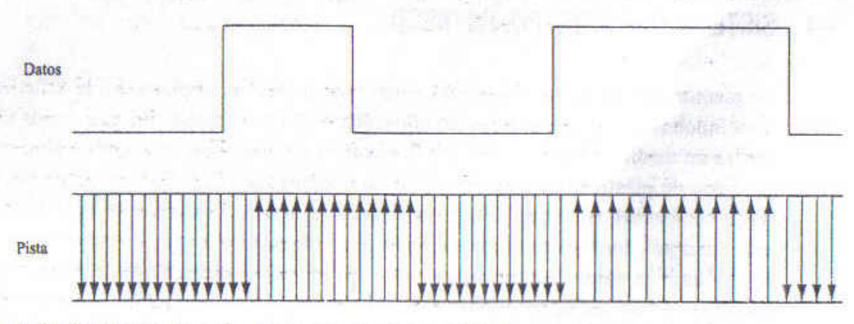


FIGURA 11–28 Técnica de grabación de no retorno a cero (NRZ).

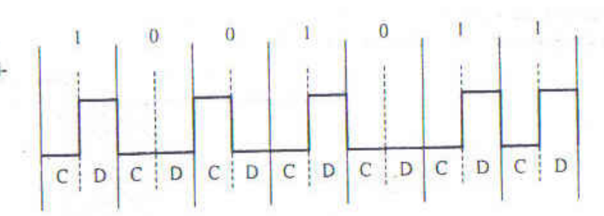


FIGURA 11-29 Modulación modificada de frecuencia (MFM) utilizada con memoria en disco.

Si está presente un pulso de datos, el tiempo de bit representa un 1 lógico. Si no hay pulsos de datos ni de reloj, el tiempo de bit representa un 0 lógico. Si está presente un pulso de reloj sin pulso de datos, el tiempo de bit también representa un 0 lógico. Las reglas cuando se almacenan datos con MFM son:

1. Un pulso de datos siempre se almacena para un 1 lógico
2. No se almacenan reloj ni datos para el primer 0 lógico en una cadena de ceros lógicos.
3. El segundo y siguientes ceros lógicos en un renglón contienen un pulso de reloj, pero no un pulso de datos.

La razón de que se intercale un pulso de reloj en el segundo y subsecuentes ceros en un renglón es mantener la sincronización cuando se leen los datos del disco. Los componentes electrónicos para recuperar los datos en la unidad de disco, tienen un ciclo de seguimiento en fase para generar un reloj y una ventana de reloj y de datos. El lazo de seguimiento de fase necesita un pulso de reloj o de datos para mantener sincronizado su funcionamiento.

Microdisco blando de 3 1/2 pulgadas. Otro tamaño muy común de disco es el de 3 1/2 pulgadas. Este disquete, en fechas recientes, se ha empezado a vender muy bien y para el futuro parece ser el que dominará. Este disquete micro es una versión mejorada del «minidisco» antes descrito. En la figura 11-30 se ilustra el disquete de 3 1/2 pulgadas.

Los diseñadores de disquetes notaron algunas desventajas en los mini, que eran una versión reducida del disco blando estándar de 8 pulgadas, poco después que apareció en el mercado. Quizá uno de los problemas más grandes con él es que está contenido en una envoltura semirrígida de plástico que se dobla con facilidad. El microdisco está alojado en una funda rígida de plástico que no se dobla o tuerce con facilidad. Con ello, se da mucha mayor protección al disco que está dentro de la funda.

Otro problema con el mini de 5 1/4 es la ranura en la cabeza que deja expuesta en forma continua la superficie del disco a los cuerpos extraños y contaminantes. Este problema también se solucionó en el micro, porque tiene una puerta de la parte superior, corrediza, accionada por resorte. La puerta permanece cerrada hasta que se introduce el disco en la unidad de disco; una vez dentro, el mecanismo desliza la puerta y deja expuesta la superficie del disco a las cabezas de lectura/escritura. Esto da mucha protección a la superficie del microdisco.

Otra importante mejora es el mecanismo deslizable, de plástico, para proteger contra escritura. En el disco mini, se colocaba una etiqueta cubriendo una muesca en un lado de la funda para impedir la escritura. Esta cinta de plástico se desprendía con facilidad dentro de las unidades de disco y ocasionaba problemas. En el micro, hay una pequeña parte corrediza, integral, de plásti-

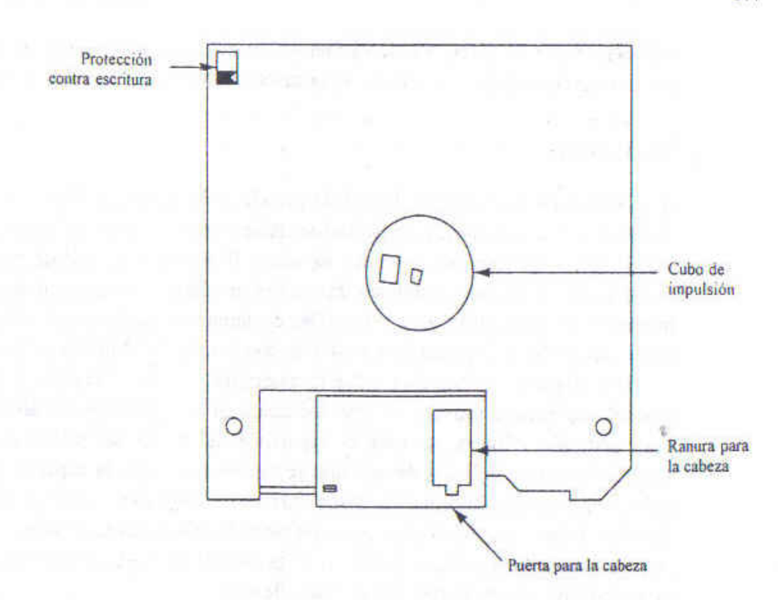


FIGURA 11-30 El microdisco blando de 3 1/2 pulgadas.

co, en vez del mecanismo que tenía la cinta de plástico. Para proteger contra escritura en el microdisquete, la pieza de plástico se mueve para *descubrir* la perforación en la funda del disco. Esto permite que llegue la luz a un sensor que inhibe la escritura.

Otra mejora más ha sido la sustitución de la perforación guía con un mecanismo diferente para el disco. Este mecanismo en el disco mini, hacía posible que la unidad de disco sujetara el disco en cualquier punto. Esto hacía necesaria la perforación guía a fin de que los componentes electrónicos pudieran encontrar el principio de una pista. La perforación guía fue otra fuente de problemas porque se acumulaba el polvo. El microdisco tiene un mecanismo de impulso con una guía que sólo permite colocarlo en una sola forma dentro de la unidad de disco. La perforación guía ya no se necesita con el nuevo mecanismo de impulso. Gracias al mecanismo de la puerta corrediza y al hecho de que no hay perforación guía, el microdisco no tiene ningún lugar en que se acumulen el polvo o la suciedad.

Hay dos tipos disponibles de microdisco: el de dos caras y doble densidad (*DSDD*) y el de alta densidad (*HD*). El microdisco de doble densidad tiene 80 pistas en cada lado y cada pista contiene 9 sectores y cada sector contiene 512 bytes de información. Esto permite tener 80 pistas por lado × 2 lados × 9 sectores × 512 bytes por sector o sean 737 280 (720K) bytes de datos que se pueden almacenar en un microdisco de doble densidad.

El microdisco de doble densidad almacena todavía más información. El disco de alta densidad tiene 80 pistas por lado, pero el número de sectores es 18, o sea el doble en cada pista. En este formato, aún se utilizan 512 bytes por sector, como en el formato de doble densidad. El número total de bytes en un microdisco de dos lados y alta densidad es de 80 pistas por lado × 2 lados × 18 sectores × 512 bytes por sector o sea 1,474, 560 (1.44 M) bytes de información.

Hace poco salió al mercado un nuevo disco de 3 1/2 pulgadas, el disco blando EHD (alta densidad extendido). Este nuevo formato almacena 2.88 Mbytes de datos en un solo disco. En la actualidad, este formato es costoso y pasará tiempo antes que se vuelva de uso común. Además,

está disponible un disco blando óptico («floptical») que almacena los datos por acción magnética con un sistema óptico de rastreo. Este disco almacena 21 Mbytes de datos.

Disco duro

Hay disponible memoria en disco más grande en la unidad de disco duro, a la cual a menudo se le llama de *disco fijo* porque no es desmontable como el disco. Al disco duro a menudo se le llama *disco rígido*. El término *unidades de disco Winchester* se utiliza para describir una unidad de disco duro. La memoria en disco duro tiene mucha mayor capacidad que la de disco flexible. La memoria en disco duro está disponible en tamaños cercanos a 1 Gbyte de datos. Los tamaños comunes, de bajo costo en la actualidad, son los de 170 Mbytes o 213 Mbytes.

Hay algunas diferencias entre la memoria en disco flexible y en disco duro. En ésta, se emplea una cabeza móvil y se leen los datos en la superficie del disco. La cabeza móvil, que es muy pequeña y ligera, no toca la superficie del disco; se mueve o «vuela» por encima de la superficie en una película de aire que se mueve junto con la superficie del disco cuando gira. La velocidad de rotación típica del disco duro es de 3000 rpm, o sea 10 veces más veloz que el disco flexible. Esta alta velocidad de rotación permite que la cabeza vuele como si fuera un avión, justo por encima de la superficie del disco. Esta característica es importante porque no hay desgaste de la superficie, como ocurre con el disco flexible.

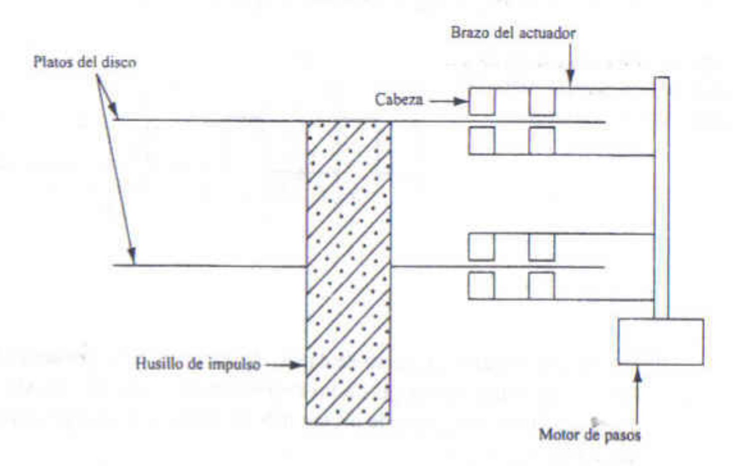
Pueden surgir problemas con las cabezas móviles y uno de ellos es la colisión. Si hay una interrupción repentina de energía, la unidad de disco duro sufre una sacudida y la cabeza puede chocar contra la superficie del disco, con lo cual se pueden dañar una o ambas. Para ayudar a evitar las colisiones, algunos fabricantes han incluido un sistema que detiene de inmediato la cabeza en una posición fija en el momento en que se interrumpe la electricidad. Este tipo de unidad de disco tiene cabezas de estacionamiento automático y al estacionarlas, se las mueve a una zona segura de reposo (un canal que no está en uso) cuando se interrumpe la corriente. Algunas unidades no tienen estacionamiento automático. Este tipo de unidad de disco suele necesitar un programa que estacione las cabezas en la pista más interna antes de que se desconecte la corriente. Esa pista más interna es un lugar seguro, porque es la última a la cual llena la unidad de disco. En este tipo de unidad de disco, el operador es quien se debe encargar del estacionamiento.

Otra diferencia entre una unidad para disco blando y una para disco duro está en el número de cabezas y superficies de discos. La unidad para disco flexible, tiene dos cabezas: una para la superficie superior y, otra, para la inferior. La unidad de disco duro puede tener hasta ocho superficies de disco (en las llamadas de cuatro platos) y hasta dos cabezas por superficie. Cada vez que se obtiene un cilindro nuevo al mover la cabeza, hay disponibles 16 nuevas pistas debajo de las cabezas. En la figura 11-31 se ilustra un sistema de disco duro.

Las cabezas se mueven de una pista a otra, con el empleo de un motor de pasos o con una bobina de voz. El motor de pasos es lento y ruidoso; el mecanismo de bobina de voz es silencioso y rápido. Para mover la cabeza, se requiere un paso por cilindro en un sistema con motor de paso a pasos para ubicar las cabezas. En un sistema en que se emplea una bobina de voz, las cabezas se pueden mover muchos cilindros con un movimiento de barrido. Con ello, la unidad de disco funciona con más rapidez cuando busque nuevos cilindros.

Otra ventaja del sistema de bobina de voz es que un servomecanismo puede vigilar la amplitud de la señal conforme sale de la cabeza de lectura y efectuar ligeros ajustes en la posición de las cabezas. Esto no es posible con un motor de pasos que funciona sólo con componentes mecánicos para ubicar las cabezas. Los mecanismos de posicionamiento de las cabezas con motor de

FIGURA 11–31 Unidad de disco duro que tiene 4 cabezas por plato.



pasos, se desalinean a menudo con el uso; el mecanismo de bobina de voz corrige cualquier desalineación.

Las unidades de disco duro pueden almacenar información en sectores de 1024 bytes de longitud. A los datos se les direcciona en un *agrupamiento* de cuatro sectores que contienen 4096 bytes en la mayor parte de las unidades de disco duro. En las unidades de disco duro se utilizan MFM o RLL para almacenar información. La MFM se describió con las unidades para disco blando. RLL (*Longitud de corrimiento limitado*) se describe en esta sección.

Una unidad de disco con MFM típica, tiene 18 sectores por pista de modo que se almacenan 18 Kbytes de datos por pista. Si una unidad de disco duro tiene capacidad de 40 Mbytes, contiene alrededor de 2280 pistas. Si la unidad de disco tiene dos cabezas, significa que contiene 1140 cilindros. Si contiene cuatro cabezas, entonces tiene 570 cilindros. Estas especificaciones varían entre unidades de disco.

Almacenamiento RLL. En las unidades de disco de longitud de corrimiento limitado (RLL), se utiliza un método diferente al de MFM para codificar los datos. El término *RLL* significa que el número de ceros en un renglón está limitado. Un método común para codificación de RLL que se emplea en la actualidad es el *RLL 2.7*, lo cual significa que el número de ceros es siempre entre 2 y 7. En la tabla 11-2 se muestra la palabra de codificación utilizada con la RLL normal.

Primero, se codifican los datos con la información de la tabla 11-2 antes de enviarlos a los circuitos electrónicos de la unidad para almacenarlos en la superficie del disco. Debido a esta técnica de codificación, es posible lograr un incremento de 50% en el almacenamiento de datos en la unidad de disco, por comparación con MFM. La diferencia principal es que la unidad de RLL tiene 27 pistas en lugar de las 18 de MFM (en algunas unidades de RLL se emplean también 35 sectores por pista).

Cabe mencionar que la codificación RLL no requiere cambio, en la mayor parte de los casos, en los componentes electrónicos de la unidad ni en la superficie de los discos. La única diferencia es una ligera reducción en el ancho de los pulsos cuando se emplea RLL, que podría requerir partículas de óxido de hierro más finas en la superficie del disco. Los fabricantes de los discos prueban la superficie y clasifican la unidad de disco como certificada para MFM o certificada

TABLA 11-2 Codificación normal de RLL 2

Datos de entrada	*Las salidas RLL*
000	000100
10	0100
010	100100
0010	00100100
11	1000
011	001000
0011	00001000

para RLL. Aparte de esta clasificación, no hay diferencia en la construcción de la unidad de disco ni en el material magnético de revestimiento de la superficie de los discos.

En la figura 11-32 se presenta una comparación de los datos de MFM y los de RLL. Se verá que el espacio que se requiere para almacenar datos en RLL se reduce por comparación con MFM. En este caso, se codifica un 101001011 para MFM y RLL, a fin de poder comparar estas dos normas. Se debe tener en cuenta que cuando se ha reducido el ancho de la señal RLL de modo que 3 pulsos se ajusten en el mismo espacio que un pulso de reloj y uno de datos en MFM. Un disco MFM de 40 Mbytes, puede contener 60 Mbytes de datos codificados en RLL. Además de contener más información, la unidad RLL se puede escribir y leer a mayor velocidad.

En todas las unidades de disco duro se utiliza codificación MFM o RLL. Hay un buen número de interfaces de unidades de disco en la actualidad. La más antigua es la interface ST-506 en la cual se emplean datos en MFM o en RLL. A un sistema de disco en que se emplee esa interface se

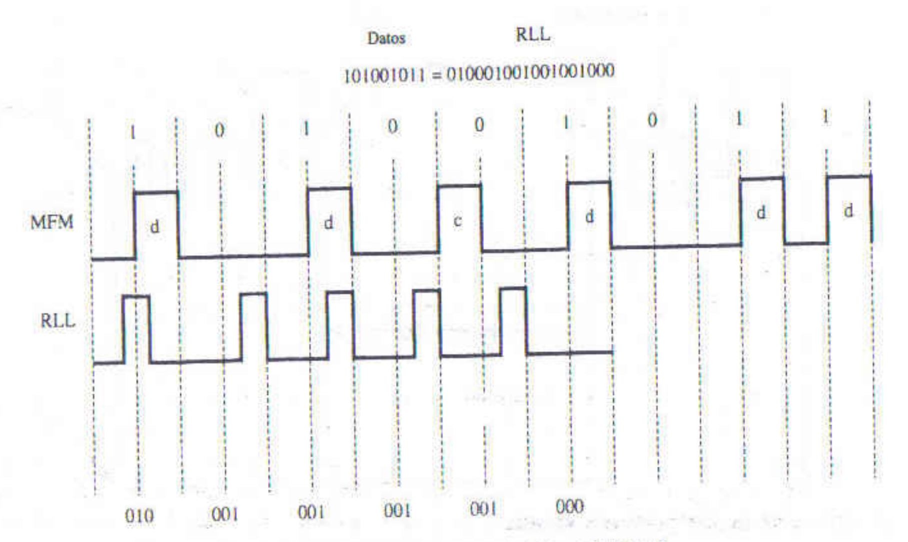


FIGURA 11-32 Comparación de MFM y RLL con el empleo de datos 101001011.

le llama también sistema de disco MFM o RLL. En la actualidad ya se utilizan las normas más recientes e incluyen ESDI, SCSI y IDE. En todas estas nuevas normas se emplea RLL aunque no se suele mencionar. La diferencia principal es la interface entre la computadora y la unidad de disco. El sistema IDE se está convirtiendo en la interface normal de memoria de disco duro.

El *sistema de interface acrecentada para disco pequeño (ESDI)* puede transferir datos entre el mismo y la computadora a velocidades cercanas a los 10 Mbytes por segundo. Una interface ST-506 se puede aproximar a una velocidad de transferencia de 860 Kbytes por segundo.

La *interface de sistema para computadoras pequeñas* (SCSI) también se utiliza porque permite conectar hasta siete discos distintos u otras interfaces con la computadora, por medio del mismo controlador de interface. EL SCSI se encuentra en algunas computadoras tipo PC y también en el sistema Apple-Macintosh. Ya ha empezado a aparecer en algunos sistemas un tipo mejorado, SCSI-II.

El sistema más nuevo es el de *componentes electrónicos integrados para la unidad (IDE)* que incluye el controlador de disco en la unidad de disco y conecta la unidad de disco con el sistema anfitrión o principal mediante un pequeño cable de interface. Esto permite conectar muchas unidades de disco a un sistema sin preocuparse de conflictos en canales o conflictos con el controlador. Las unidades IDE se encuentran en los más recientes sistemas IBM PS-2 y en muchos clones. La interface IDE también puede manejar otros dispositivos de E/S además del disco duro. Esta interface, por lo general, incluye una memoria caché de 32 Kbytes para los datos del disco. La caché acelera las transferencias del disco. Los tiempos comunes de acceso para la unidad IDE, a menudo, son menores de 12 ms, mientras que el tiempo de acceso para un disco flexible es de 200 ms.

Disco óptico

La memoria en disco óptico (véase figura 11-33) suele estar disponible en dos formas: *CDROM* (memoria de sólo lectura en disco compacto) y la *WORM* (escribir una vez, casi siempre leer). CDROM es el disco óptico de más bajo costo, pero adolece de falta de velocidad. Los tiempos de acceso típicos de la CDROM son de 300 ms o más, o sea casi lo mismo que en un disco flexible. (Se debe tener en cuenta que hay dispositivos CDROM más lentos y hay que evitarlos.) La memoria magnética en el disco duro puede tener tiempos de acceso tan pequeños como 16 ms. El CDROM también adolece de falta de programas de aplicación en este momento. El CDROM está disponible para almacenamiento de alto volumen de datos, como la Biblia, enciclopedias, artículos de revistas, etcétera. Ninguna de estas aplicaciones es de gran atractivo a los precios actuales. En el futuro, si se reducen los tiempos de acceso y se introducen más aplicaciones, el CDROM se usará mucho más. Una CDROM almacena 660 Mbytes de datos o una combinación de datos y pasajes musicales. Conforme se perfeccionan los sistemas y adquieren mayor actividad visual, el empleo de la unidad CDROM se volverá más común.

La unidad WORM ha de poder lograr muchas más aplicaciones comerciales que el CDROM. El problema es que esta aplicación es muy especializada debido a la naturaleza del WORM. Debido a que los datos sólo se pueden escribir una vez, la aplicación principal es en la industria de la banca, compañías de seguros y muchas otras organizaciones que tienen almacenamiento masivo de datos. El WORM se suele utilizar para formar una cadena de transacciones para su auditoría, que se almacenan en forma temporal en WORM y sólo se recuperan durante la auditoría. Se podría decir que WORM es un dispositivo para archivo temporal.

Muchos sistemas de memoria WORM y de lectura y escritura en disco óptico, tienen interface con el microprocesador bajo las normas SCSI o ESDI para interface utilizadas con la memoria en

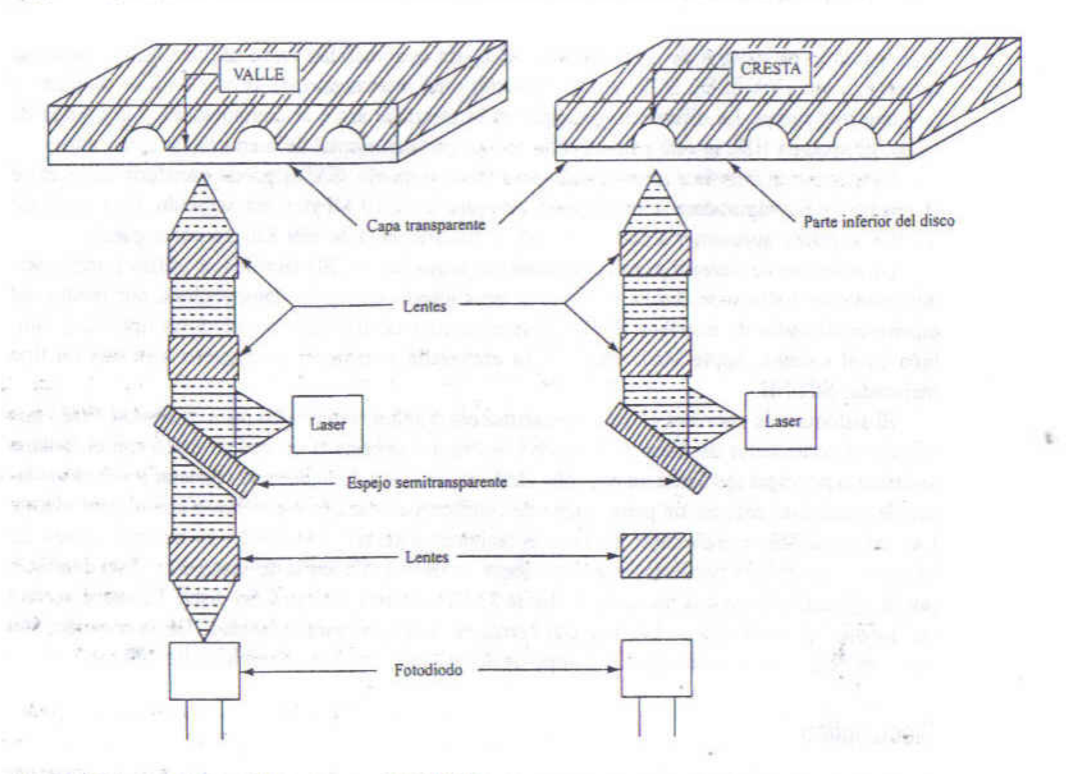


FIGURA 11-33 El sistema típico de memoria CDROM.

disco duro. La diferencia es que las unidades actuales para disco óptico no son más veloces que la mayor parte de las de disco flexible. Algunas unidades CDROM tienen interface con el microprocesador mediante interfaces patentadas que no son compatibles con otras unidades de disco.

La ventaja principal del disco óptico es su durabilidad. Debido a que se utiliza un rayo laser de estado sólido para leer los datos en el disco, y el punto focal está debajo de una capa protectora de plástico, pueden haber raspaduras pequeñas y partículas de polvo o suciedad en la superficie del disco y de todos modos se podrá leer en forma correcta. Esta característica hace posible que la unidad de disco óptico requiera menos atención que la de disco flexible. Casi la única forma de destruir los datos en un disco óptico es despedazarlo o inferirle rayaduras muy profundas.

11-5 SISTEMAS DE VIDEO

Las unidades de exhibición de video (monitor) actuales provienen de un fabricante de equipo original (*OEM*) al cual se le compran para incluirlos en un sistema. En la actualidad, hay muchos tipos de monitores de video, entre los cuales se pueden escoger a color o monocromáticos.

En los monitores monocromáticos se suele exhibir la información contra un fondo ámbar, verde o blanco como papel; esta última cada vez se emplea más. Las aplicaciones más comunes son para tenerlos sobre el escritorio para revisión o edición y para dibujo asistido por computadora (CAD).

Los monitores de color son más variados. Hay algunos disponibles que aceptan información en una señal de video compuesta, en forma muy similar a un televisor doméstico, como señales con valor de voltaje de 0 o 5 volts TTL y como señales analógicas. Los monitores del tipo de televisor van desapareciendo, porque su resolución es muy baja. Hoy en día, en muchas aplicaciones se requieren gráficos de alta resolución que no se pueden exhibir en un monitor del tipo de televisor doméstico. Los primeros monitores del tipo de TV doméstica se emplearon en Commodore 64, Apple 2 y sistemas similares de computadoras.

Señales de video

En la figura 11-34 se ilustra la señal de video compuesto. La señal consta de cierto número de partes que se requieren para este tipo de exhibición. Se debe tener en cuenta que estas señales incluyen no sólo video, sino también pulsos de sincronización, pedestales de sincronización y una ráfaga de color. También se verá que no se incluye audio con la señal de video compuesta, porque el audio se produce en la computadora y se le da salida desde una bocina o altavoz en el gabinete de la computadora. La principal desventaja del monitor con video compuesto son la resolución y las limitaciones en el color. La principal desventaja de la exhibición de video compuesto son la resolución y las limitaciones en el color. Las señales de video compuestas están destinadas a emular las señales de video para los televisores, a fin de que un televisor doméstico pudiera funcionar como monitor de video.

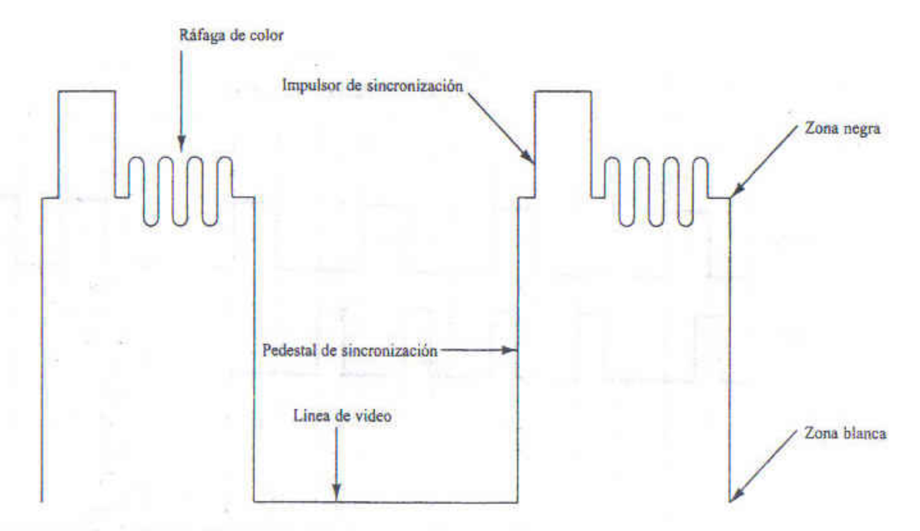


FIGURA 11-34 La señal de video compuesta.

En la mayor parte de los sistemas modernos de video, se emplean señales directas de video que se generan con señales de sincronización separadas. En un sistema de video directo, la información de video se pasa al monitor por medio de un cable que tiene líneas separadas para video y pulsos de sincronización. Recuerde que estas señales se combinaron en la señal de video compuesta.

En el monitor monocromático se utiliza un conductor para el video, uno para sincronización horizontal y uno para sincronización vertical. A menudo suelen ser los únicos alambres de señal que tienen. En un monitor de color se emplean tres señales de video: una representa el rojo, otra el verde y la tercera el azul. A estos monitores, se les llama con frecuencia RGB por las iniciales en inglés de los colores primarios de la luz para el video, que son: rojo (R), verde (G) y azul (B).

El monitor TTL RGB

El monitor RGB está disponible como analógico o TTL. En el monitor TTL se emplean señales de valor TTL (0 o 5 volts) como entradas de video y una cuarta línea llamada intensidad para poder cambiar ésta. El monitor TTL RGB puede exhibir un total de 16 colores diferentes y se emplea en el sistema CGA (adaptador gráfico a color) que se empleaba en sistemas antiguos de computadoras.

En la tabla 11-3 se enumeran estos 16 colores y también las señales de TTL presentes para generarlos. Ocho de los 16 colores se generan con alta intensidad y los otros ocho con baja intensidad. Los tres colores del video son rojo, verde y azul, que son los colores primarios de la luz. Los colores secundarios son cyan o azulado, magenta y amarillo. El azulado es una combinación de señales azules y verdes y su tonalidad es azul verdoso. El magenta es una combinación de señales azules y rojas y su color es púrpura o morado. Son combinaciones de las señales de video rojo y verde. Si se desean colores adicionales no se suele emplear el video TTL. Se produjo un método con el empleo de señales de video TTL para color mediano y bajo, con las que se lograron 32 colores, pero nunca tuvo gran aceptación en el campo.

TABLA 11–3 Los 16 colores disponibles en una exhibición (CGA) de TTL RGB

Rojo	*Verde*	*Azul*	*Intensidad*	Color
0	0	0	0	Negro
0	0	0	1	Gris
0	0	1	0	Azul claro
0	0	1	1	Azul
0	1	0	0	Verde claro
0	1	0	1	Verde
0	1	1	0	Azulado claro
0	1	1	1	Azulado
1	0	0	0	Rojo claro
1	0	0	1	Rojo
1	0	1	0	Magenta claro
1	0	1	1	Magenta
1	1	0	0	Café
1	1	0	1	Amarillo
1	1	1	0	Blanco
1	1	1	1	Blanco intenso

En la figura 11-35 se ilustra el conector que se emplea más a menudo en un monitor TTL RGB o en un TTL monocromático. Ese conector tiene 9 terminales. Dos de ellas se emplean para tierra, tres para el video, dos para señales de sincronización o de retrazo del haz y una para la intensidad. Se verá que la terminal 7 está marcada video normal y es la que se emplea en un monitor monocromático para la señal de brillantez. En los monitores TTL monocromáticos se emplea el mismo conector de 9 terminales que en los TTL RGB.

El monitor RGB analógico

Para exhibir más de 16 colores, se necesita un monitor analógico. A éstos se les suele denominar como monitores RGB analógicos. Estos monitores todavía tienen las tres señales de entrada de video, pero no tienen entrada para intensidad. Debido a que las señales de video son analógicas en vez de TTL de dos valores, tienen cualquier valor de voltaje entre 0.0 y 0.7 volts. Esto permite exhibir un número infinito de colores porque se puede generar un número infinito de valores de voltaje entre el mínimo y el máximo. En la práctica, se genera un número finito de valores. Suelen ser 256K, 16M o 24M colores, según sea la norma.

En la figura 11-36 se ilustra el conector utilizado para un monitor RGB analógico o un monocromático analógico. Se verá que el conector tiene 15 terminales y soporta los monitores RGB analógicos y monocromáticos. La forma en que se exhiban los datos en un monitor RGB analógico, depende de la norma de la interface utilizada con el monitor. La terminal 9 es una clave que significa que no hay hembra en el conector para esta terminal.

En casi todos los monitores analógicos se utiliza un *convertidor de digital/analógico* (DAC) para generar cada voltaje de color de video. Un estándar común utiliza un DAC de 6 bits, para que cada señal de video genere 64 diferentes valores de voltaje entre 0.0 y 0.7 volts. Hay 64 diferentes valores de rojo, 64 distintos para verde y 64 diferentes para azul. Esto permite exhibir 64 × 64 × 64 diferentes colores o sea 262 144 (256K) colores.

Hay otras disposiciones posibles, pero la velocidad del DAC es crítica. Casi todos lo monitores modernos requieren un tiempo de conversión de funcionamiento máximo entre 25 ns y 40 ns. Cuando se perfeccione la tecnología de los convertidores, estará disponible una resolución adicional a un precio razonable. Si se utilizan convertidores de 7 bits para generar las señales de video, se exhiben 128 × 128 × 128 colores o sea 2 097 152 (2M) colores. En este sistema, se

FIGURA 11–35 El conector de 9 terminales utilizado en un monitor TTL.

DB9

5 9 4 8 3 7 2 6 1

Terminal	Función
1	Tierra
2	Tierra
3	Video rojo
4	Video verde
5	Video azul
6	Intensidad
7	Video normal
8	Retraza horizontal
9	Retraza vertical

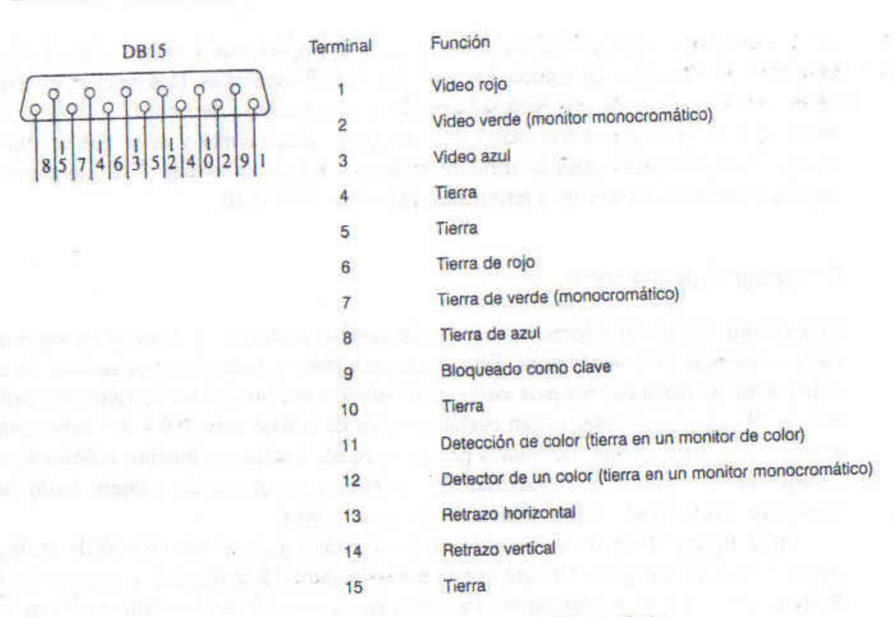


FIGURA 11–36 El conector de 15 terminales que se encuentra en un monitor analógico.

necesita un código de color de 21 bits a fin de aplicar un código de 7 bits a cada DAC. También se emplean convertidores de 8 bits y permiten 256 × 256 × 256 o sea 16 777 216 (16M) colores.

En la figura 11-37 se ilustra el circuito de generación de video utilizado en muchos estándares comunes para video como EGA (adaptador para gráfico mejorado) y VGA (arreglo gráfico variable), utilizados en una IBM PC. Este circuito se utiliza para generar video VGA. Se verá que cada color se genera con un código digital de 18 bits. Seis de los 18 bits se emplean para generar cada voltaje de color cuando se aplican a las entradas de un DAC de 6 bits.

Se emplea para la paleta una SRAM de alta velocidad (tiempo de acceso de menos de 40 ns) para almacenar 256 diferentes códigos de 18 bits, que representan 256 distintos tonos o matices. Este código de 18 bits se aplica a los DAC. La entrada de dirección a la SRAM selecciona uno de los 256 colores almacenados como códigos binarios de 18 bits. Este sistema permite exhibir, cada vez, 256 colores de los 256K posibles. Para seleccionar cualquiera de los 256 colores, se emplea un código de 8 bits almacenado en la RAM de video para especificar el color de un elemento de una imagen. Si se utilizan más colores en un sistema, el código debe ser más ancho. Por ejemplo, un sistema que exhibe 1024 de los 256K colores requiere un código de 10 bits para direccionar a la DRAM que contiene 1024 localidades y cada una contiene un código de color de 18 bits.

En la Apple Macintosh IIci se emplea un código binario de 24 bits para especificar cada color en su adaptador de video a color. Cada DAC tiene 8 bits de ancho, lo cual significa que cada convertidor puede generar 256 diferentes valores de voltaje de video. Hay 256 × 256 × 256 o sea un total de 16 777 2156 colores diferentes posibles. Igual que con el estándar IBM VGA, sólo se exhiben 256 colores cada vez. La SRAM en la interface de Apple es de 256 × 24 en vez de 256 × 18.

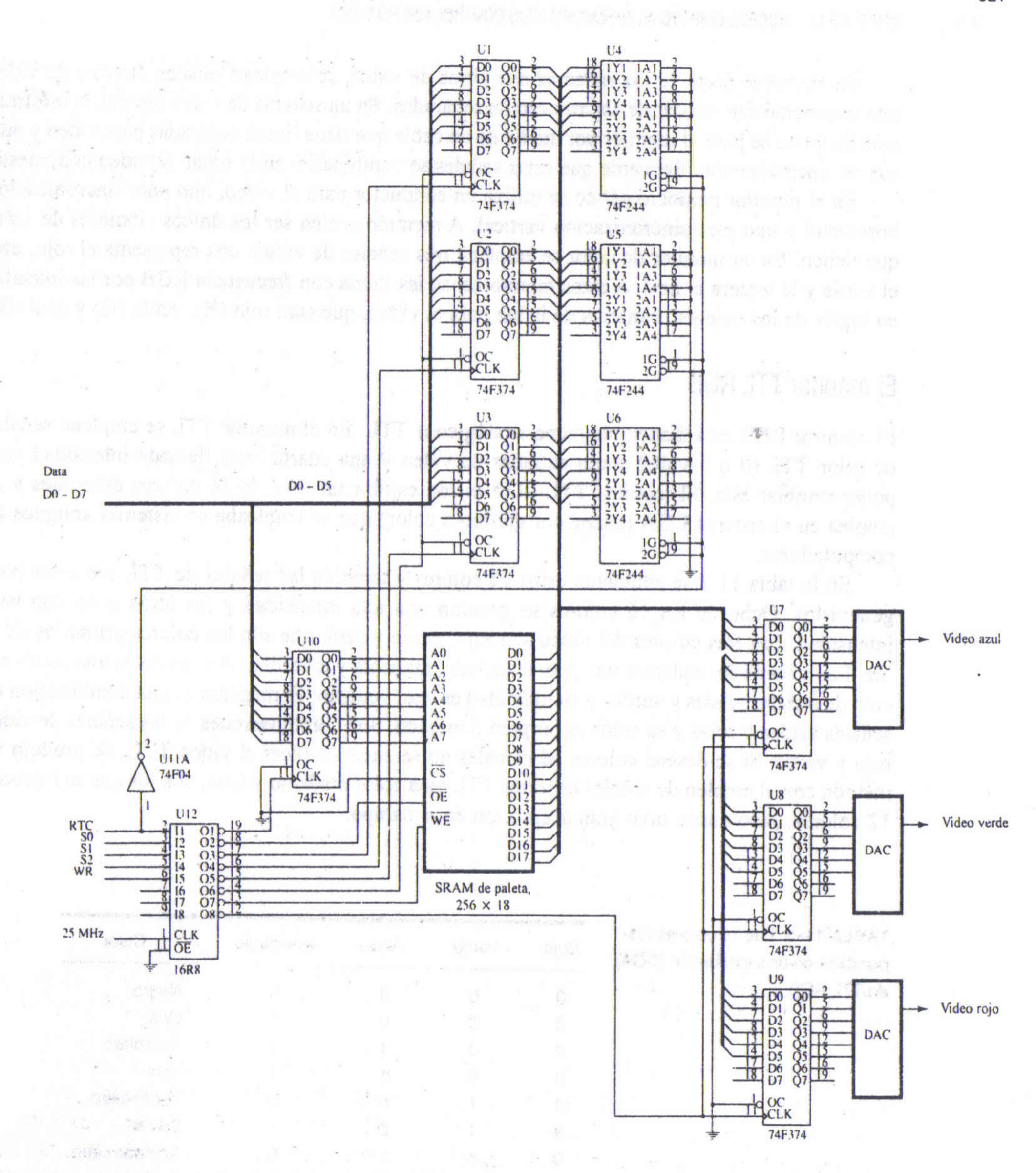


FIGURA 11–37 Generación de señales de video VGA.

Siempre que se coloca un color en el monitor, en la inteligencia de que RTC es un 0 lógico, el sistema envía el código de 8 bits que representa un color a las conexiones de D0-D7. Luego, el PAL 16R8 genera un pulso de reloj para U10 que captura el código de color. Después de 40 ns (un pulso de reloj de 25 MHz), el PAL genera un pulso de reloj para los registros del DAC (U7, U8

y U9). Esta cantidad de tiempo se requiere para que la SRAM de la paleta pueda consultar el contenido de 18 bits de la localidad de memoria seleccionada por U10. Una vez que el código de color de 18 bits se captura en U7-U9, los tres DAC lo convierten a tres voltaje de video para el monitor. Este proceso se repite para cada elemento de imagen (*pixel*) que se exhibe de 40 ns de ancho que se exhibe. El pixel es de 40 ns de ancho porque en estè sistema se emplea un reloj de 25 MHz. Se puede logrär una resolución más alta en este sistema si se emplea una frecuencia de reloj más alta.

Si hay que cambiar los códigos de color de 18 bits almacenados en la SRAM, siempre se logra durante el retrazo cuando RTC es un 1 lógico. Esto evita que cualquier ruido del video altere la imagen exhibida en el monitor.

Para cambiar un color, en el sistema se utilizan las entradas S0, S1 y S2 al PAL para seleccionar U1, U2, U3 o U10. Primero, se envía la dirección del color que se va a cambiar al registro U10. Esto direcciona a una localidad en la SRAM de la paleta. Luego, se carga cada nuevo color de video en U1, U2 y U3. A continuación el PAL genera un pulso de escritura para la entrada $\overline{WE}$ a la SRAM para escribir el código del nuevo color en la SRAM de la paleta.

El retrazo o retorno ocurre 70.1 veces por segundo en el sentido vertical y 31 500 veces por segundo en el sentido horizontal en un modo de video de 640 × 480. Durante el retrazo, el voltaje de la señal de video enviado al monitor debe ser cero. Esto hace que se exhiba negro y no se observe el retrazo. El retrazo se emplea también para mover el haz de electrones a la esquina superior izquierda para el retrazo vertical y al margen izquierdo de la pantalla para el retrazo horizontal.

El circuito ilustrado hace que los registros U4-U6 se habiliten con lo cual aplican 00000 cada uno en el registro del DAC durante el retrazo. Los registros del DAC capturan este código y generan 0 volt por cada señal de color, para apagar la pantalla. Por definición, 0 volt se considera como el valor de negro para el video y 0.7 volts se considera como plena intensidad en una señal de color.

La resolución de la exhibición, por ejemplo, 640 × 400 determina la cantidad de memoria que se requiere para la tarjeta de interface del video. Si esta resolución se emplea con una exhibición de 256 colores (8 bits por pixel), entonces se necesitan 640 × 400 bits (256 000) de memoria para almacenar todos los pixeles para la exhibición. Se pueden lograr exhibiciones con resoluciones más altas, pero cualquiera puede suponer que se necesitará más memoria. Una exhibición de 640 × 400 tiene 400 líneas de rastreo y 640 pixeles por línea. Una **línea de rastreo** es la línea horizontal de información que se exhibe en el monitor. Un pixel es la subdivisión más pequeña de esta línea horizontal.

En la figura 11-38 se ilustra la exhibición de video y se muestran las líneas de video y el retrazo. En esta ilustración la inclinación de cada línea de video está muy exagerada, igual que lo está el espacio entre líneas. Se ilustra el retrazo en sentido vertical y horizontal. En el caso de una exhibición VGA, como se describió, el retrazo vertical ocurre 70.1 veces por minuto exactamente y el retrazo horizontal ocurre 31 500 veces por segundo exactamente. (En la Apple Macintosh IIci se emplea una frecuencia vertical de 66.6 Hz y una velocidad horizontal de 35 kHz para generar una exhibición de color de 640 × 400.)

Para poder generar 640 pixeles a lo largo de una línea, se necesitan 40 ns × 640 o sea 25.6 microsegundos. Una frecuencia horizontal de 31 500 Hz permite un tiempo por línea horizontal de 1/31 500 igual a 31.746 microsegundos. La diferencia entre estos dos tiempos es el tiempo para el retrazo que se le permite al monitor. (La Apple Macintosh IIci tiene un tiempo de línea horizontal de 28.57 microsegundos.)

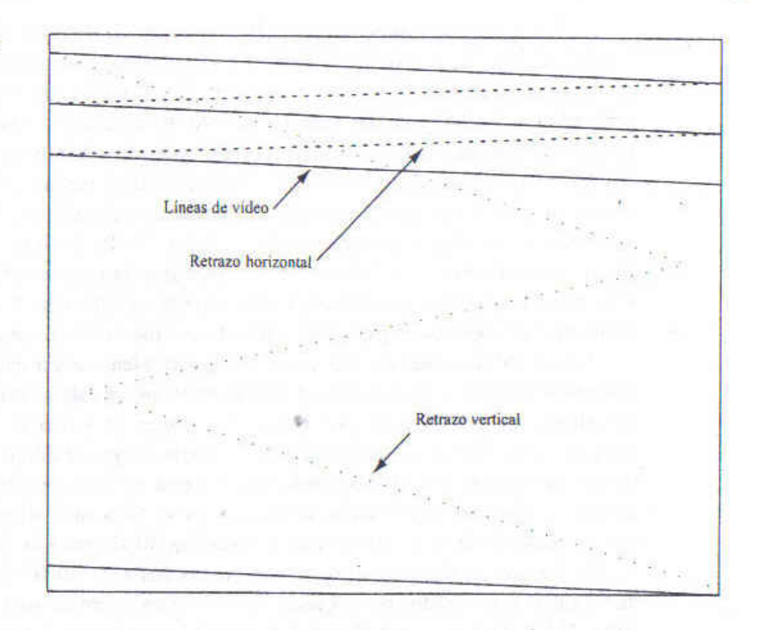


FIGURA 11-38 Una pantalla de video y se ilustran las líneas de trama y de los retrazos.

Debido a que la frecuencia de repetición del retrazo vertical es de 70.1 Hz, para determinar el número de líneas generadas se divide el tiempo vertical entre el tiempo horizontal. [En el caso de una exhibición de 640 × 400 en VGA, son 449.358 líneas.] Sólo se utilizan 400 de estas líneas para exhibir información, pues el resto se pierde durante el retrazo. Dado que se pierden 49.358 líneas durante el retrazo, el tiempo para ello es de 49.358 × 31 766 microsegundos o 1 568 microsegundos. Durante este tiempo más o menos largo es cuando se cambia la SRAM de paleta de colores o se actualiza el sistema de memoria de video para una nueva exhibición. En la Apple Macintosh IIci (640 × 480), se generan 525 líneas. De ese número total, se pierden 45 líneas durante el retrazo vertical.

Otras resoluciones para exhibición son 800 × 600 y 1024 × 768. La exhibición SVGA (super VGA) de 800 × 600 es ideal para un monitor de color de 14 pulgadas, mientras que la EVGA o XVGA (VGA extendida) es ideal para los monitores de 21 o de 25 pulgadas que se utilizan con los sistemas CAD. Estas resoluciones parecen ser sólo números, pero se debe tener en cuenta que un televisor doméstico promedio tiene una resolución de alrededor de 400 × 300. La exhibición de alta resolución disponible en los sistemas de computadora es mucho más clara que en un televisor doméstico. Una resolución de 1024 × 768 se aproxima a la que hay en una película de 35 mm. La única desventaja de la exhibición de video en el monitor de computadora, es el número de colores que se exhiben cada vez, pero conforme pasa el tiempo, es seguro que mejore. Los colores adicionales permiten que la imagen aparezca más real, debido al sutil sombreado que se requiere para tener una imagen de alta calidad y casi real.

Si el sistema de exhibición funciona con una velocidad vertical de 60 Hz y un tiempo horizontal de 15 600 Hz, el número de líneas generadas será de 15 600/60 o 260 líneas. El número más probable de líneas útiles será de 240, porque se pierden 20 durante el retrazo vertical. Es evidente que para ajustar el número de líneas de exploración se cambian las frecuencias de rastreo vertical y horizontal.

La frecuencia de rastreo vertical debe ser de 50 Hz o mayor o habrá parpadeo. La frecuencia vertical no debe ser mayor de alrededor de 75 Hz porque pueden ocurrir problemas con la bobina de deflexión vertical. El haz de electrones en un monitor se ubica con un campo magnético generado por las bobinas que rodean el cuello del cinescopio. Dado que el campo magnético se genera con bobinas, la frecuencia de la señal aplicada a la bobina es limitada.

La frecuencia de rastreo horizontal también está limitada por la construcción de las bobinas del yugo. Por ello, es normal encontrar la frecuencia aplicada a las bobinas horizontales dentro de un intervalo estrecho, que suele ser de 30 000 a 37 000 Hz o de 15 000 a 17 000 Hz. Algunos de los monitores más modernos se llaman multisync porque la bobina de deflexión está bobinada a fin de poder excitarla con diferentes frecuencias de deflexión A veces, las bobinas de vertical y horizontal son bobinadas para tener diferentes velocidades de rastreo vertical y horizontal.

En las exhibiciones de alta resolución se emplea rastreo entrelazado o no entrelazado. El sistema de rastreo no entrelazado se emplea en todos los estándares, excepto los más elevados. En el sistema con entrelazado, para exhibir la imagen de video se dibuja primero la mitad de la imagen con las líneas de rastreo de número impar; luego, se dibuja la otra mitad con las líneas de rastreo de número par. Por supuesto este sistema es más complejo y de mayor eficiencia sólo porque las frecuencias de rastreo se reducen en un 50% en el sistema con entrelazado. Por ejemplo, un sistema de video en el cual se emplean 60 Hz para la frecuencia de rastreo vertical y 15 720 Hz para la frecuencia horizontal, genera 262 (15 720/60) líneas de video a una velocidad de 60 cuadros completos por segundo. Si se hace un ligero cambio a 15 750 Hz, se generan 262.5 líneas (15 750/60) por lo cual se requieren dos barridos completos para dibujar una imagen completa de 525 líneas de video. Se verá que un ligero cambio en la frecuencia horizontal, duplicó el número de líneas de rastreo.

11–6 RESUMEN

1. La entrada HOLD se emplea para solicitar una acción de DMA y la salida HLDA señala que la sección del canal del sistema está en vigor. Cuando se pone un 1 lógico en la entrada HOLD, entonces el microprocesador: (1) para la ejecución del programa, (2) pone sus canales de direcciones, datos y control en su estado de alta impedancia y (3) señala que la sección del canal del sistema está en vigor, poniendo un 1 lógico en la terminal HLDA.
2. Una operación de lectura de DMA transfiere datos desde una localidad en la memoria hasta un dispositivo externo de E/S. Una operación de escritura de DMA transfiere datos desde el E/S hasta la memoria. También está disponible una transferencia de memoria a memoria que permite transferir datos entre dos localidades de memoria, con las técnicas de DMA.
3. El controlador de acceso directo a memoria 8237 (DMA) tiene cuatro canales que se pueden ampliar para incluir canales adicionales de DMA.
4. La memoria en disco es en forma de almacenamiento en disco flexible que son el mini de 5 1/4 pulgadas y el micro de 3 1/2 pulgadas. Ambos discos son de los tipos de dos lados y doble densidad (DSDD) o de alta densidad (HD). El disco DSDD de 5 1/4 pulgadas almacena 360 Kbytes de datos y el HD de 5 1/4 pulgadas almacena 1.2 Mbytes de datos. El disco DSDD de 3 1/2 pulgadas almacena 720 Kbytes de datos y el HD de 3 1/2 almacena 1.44 Mbytes de datos.

5. Los datos se almacenan en la memoria de disco blando con grabación NRZ (no retorno a cero). Con este método, se satura el disco con energía magnética de una polaridad para un 1 lógico y con la polaridad opuesta para un 0 lógico En cualquier caso, el campo magnético nunca retorna a cero. Con esta técnica, no hay necesidad de tener una cabeza borradora separada.
6. Los datos se graban en los discos con métodos de codificación de modulación de frecuencia modificada (MFM) o de longitud de corrimiento limitado, RLL. Con el método MFM se graba un pulso de datos para un 1 lógico, ningún datos o pulsos de reloj para el primer 0 lógico en una cadena de ceros y un pulso de reloj para el segundo y subsecuentes ceros lógicos en una cadena de ceros. El método RLL codifica los datos de modo de poder empacar 50% más información en la misma superficie de disco. En casi todos los sistemas modernos de memoria en disco se emplea el método de codificación RLL.
7. Los monitores de video son TTL o analógicos. En el TTL se utilizan dos valores discretos de voltaje: 0 volt y 5.0 volts. En el monitor analógico se utiliza un número infinito de valores de voltaje entre 0 y 0.7 volt. En el monitor analógico se puede exhibir un número infinito de valores de video; el TTL está limitado a dos valores de video.
8. El monitor TTL para color exhibe 16 colores diferentes. Se logra con tres señales diferentes de video (rojo, verde y azul) y una entrada para intensidad. El monitor analógico para color puede exhibir un número infinito de colores mediante sus tres entradas de video. En la práctica, la forma más común del sistema VGA puede exhibir 256 Kcolores.
9. Los estándares actuales para video incluyen VGA (640 × 480), SVGA (800 × 600) y EVGA o XVGA (1024 × 768). En los tres casos, la información de video puede ser de 256 de los 256 K colores posibles.

11-7 CUESTIONARIO Y PROBLEMAS

1. ¿Qué terminales del microprocesador se emplean para solicitar y reconocer una transferencia de DMA?
2. Explique lo que ocurre cuando se pone un 1 lógico en la terminal de entrada HOLD.
3. Una lectura DMA transfiere datos desde________hasta ______.
4. Una escritura DMA transfiere datos desde ________hasta ________.
5. ¿Con cuáles señales de canales selecciona el controlador de DMA la localidad de la memoria utilizada para una transferencia de DMA?
6. ¿Mediante cuál terminal selecciona el controlador de DMA el dispositivo de entrada/salida que se utiliza durante una transferencia de DMA?
7. ¿Qué es una transferencia de DMA de memoria a memoria?
8. Describa el efecto en el microprocesador y el controlador de DMA cuando las terminales HOLD y HLDA están a sus valores de 1 lógico.
9. Describa el efecto en el microprocesador y el controlador de DMA cuando las terminales HOLD y HLDA están a sus valores de 0 lógico.
10. El controlador 8237 para DMA tiene _____canales.
11. Si se decodifica el controlador 8237 de DMA en los puertos de E/S 2000H-200FH, ¿cuáles puertos se utilizan para programar el canal 1?

12. ¿Cuál registro del controlador 8237 de DMA se programa para inicializar el controlador?
13. ¿Cuántos bytes se transfieren con el controlador 8237 de DMA?
14. Escriba una secuencia de instrucciones que transfieran datos desde la localidad de memoria 210000H-210FFH, hasta 20000H-200FFH con el canal 2 del controlador de DMA 8237. Hay que inicializar el 8237 y emplear el registro descrito en la sección 11-1 para retener A19-A-16.
15. Escriba una secuencia de instrucciones que transfieran datos desde la memoria hasta un dispositivo de E/S externo con el empleo del canal 3 del 8237. La zona de memoria que se va a transferir está en la localidad 20000H-20FFFH.
16. Al disco de 5 1/4 pulgadas se le llama __________.
17. Al disco de 3 1/2 pulgadas se le llama ____________.
18. Los datos se graban en anillos concéntricos en la superficie de un disco que se conoce como__________.
19. Una pista está dividida en secciones de datos llamadas__________.
20. En un disco de dos lados, las pistas superior e inferior, juntas, se llaman__________.
21. ¿Por qué se utiliza una grabación NRZ en un sistema de memoria en disco?
22. Dibuje el diagrama de temporización generado para escribir un 1001010000 con codificación MFM.
23. Dibuje el diagrama de temporización generado para escribir un 1001010000 con codificación RLL.
24. ¿Qué es una cabeza móvil?
25. ¿Por qué se deben detener o estacionar las cabezas en un disco duro?
26. ¿Cuál es la diferencia entre un mecanismo de localidad de cabezas con bobina de voz y un mecanismo de localidad de cabezas con motor de pasos?
27. ¿Qué es un WORM?
28. ¿Qué es un CDROM?
29. ¿Cuál es la diferencia entre un monitor TTL y uno analógico?
30. ¿Cuáles son los tres colores primarios de la luz?
31. ¿Cuáles son los tres colores secundarios de la luz?
32. ¿Qué es un pixel?
33. Una exhibición de video con una resolución de 800 × 600 contiene __________ líneas de información de video, cada una divida en _______pixeles.
34. Explique cómo se pueden exhibir 16 colores distintos en un monitor TTL RGB.
35. Explique la forma en que un monitor analógico RGB puede exhibir un número infinito de colores.
36. Si en un sistema de video analógico RGB se emplea un DAC de 7 bits, se pueden generar ________ colores diferentes.
37. ¿Por qué el estándar VGA permite exhibir, de una sola vez, 256 colores diferentes de los 256 K disponibles?
38. Si en un sistema de video se emplea una frecuencia vertical de 60 Hz y una frecuencia horizontal de 32 400 Hz, ¿cuántas líneas de rastreo se generan?

CAPITULO 12

La familia de coprocesadores aritméticos

INTRODUCCION

La familia Intel de coprocesadores aritméticos incluye los 8087, 80287, 80387SX, 80387DX, 80487SX y el microprocesador 80486DX, que contiene su propio coprocesador aritmético integrado. El conjunto de instrucciones y la programación para estos dispositivos es casi idéntica; la diferencia consiste en que cada coprocesador está diseñado para funcionar con un microprocesador Intel diferente. Este capítulo proporciona los detalles genéricos sobre toda la familia.

La familia del coprocesador, que identificaremos como 80X87, puede multiplicar, dividir, sumar, restar y calcular raíz cuadrada, tangente parcial, arco tangente parcial, y logaritmos. Los tipos de datos incluyen: números enteros con signo de 16, 32, y 64 bits; datos en BCD de 18 dígitos y números con el punto decimal flotante de 32, 64, y 80 bits. Las operaciones realizadas por el 80X87 por lo general se ejecutan aproximadamente 100 veces más rápidas que otras operaciones equivalentes escritas con los programas más eficientes.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Convertir datos entre decimales y el tipo de datos permitido para el coprocesador aritmético.
2. Explicar la operación del coprocesador aritmético 80X87 cuando está conectado al microprocesador.
3. Explicar la operación y los modos de direccionamiento para cada instrucción del coprocesador aritmético.
4. Desarrollar programas para resolver problemas complejos de aritmética utilizando el coprocesador aritmético.

12–1 FORMATOS DE DATOS PARA EL COPROCESADOR ARITMETICO

Esta sección del texto presenta los tipos de datos utilizados con todos los miembros de la familia (8087, 80287, 80387SX, 80387DX, 80487SX, y 80486DX). Estos tipos incluyen los números enteros con signo, BCD, y el punto decimal flotante. Cada uno tiene un propósito específico dentro de un sistema, y muchos sistemas requieren de los tres tipos de datos.

Números enteros con signo

Los números enteros con signo utilizados con el coprocesador son básicamente los mismos a los descritos en el capítulo 1. Los números enteros con signo, cuando se utilizan con el coprocesador aritmético, son de 16 (palabra), de 32 (el número entero corto), o de 64 bits de ancho (el número entero largo). La conversión entre el formato de decimal y el formato de número entero con signo, se maneja exactamente de la misma forma que para los números enteros con signo de 8 bits descritos en el capítulo 1. Como se recordará, los números positivos se guardan en forma real con el signo en el bit más a la izquierda como 0, y los números negativos se guardan en forma de complemento a dos con el signo en el bit más a la izquierda como un 1.

Los números enteros de palabra varían en valor de –32,768 a + 32,767, el número entero corto de $-2 \times 10^{+9}$ a $+ 2 \times 10^{-9}$, y el número entero largo de $-9 \times 10^{+18}$ a $+ 9 \times 10^{-18}$. Los tipos de datos de números enteros se encuentran en aplicaciones que utilizan el coprocesador aritmético. Véase la figura 12-1 que muestra estas tres formas de datos enteros con el signo.

Los datos se almacenan en la memoria usando los mismos directivos del ensamblador descritas y utilizadas en capítulos anteriores. Se utiliza el directivo DW para definir palabras, DD para definir números enteros cortos, y la DQ para definir números enteros largos. El ejemplo 12-1 muestra cómo varios tamaños diferentes de números enteros con signo se definen para su uso por el ensamblador.

EJEMPLO 12–1

```
0000 0002                DATO1   DW   +2       ;número entero de 16 bits
0002 FFDE                DATO2   DW   -34      ;número entero de 16 bits
0004 000004D2            DATO3   DD   +1234    ;número entero corto
0008 FFFFFF9C            DATO4   DD   -100     ;número entero corto
000C 0000000000005BA0    DATO5   DQ   +23456   ;número entero largo
0014 FFFFFFFFFFFFFF86    DATO6   DQ   -122     ;número entero largo
```

Forma decimal codificado en binario (BCD)

La forma decimal codificado en binario (BCD) requiere 80 bits de memoria. Cada número se guarda como un número entero empacado de 18 dígitos en 9 bytes de memoria y con dos dígitos por byte. El décimo byte contiene solamente un bit de signo para el número BCD con signo de 18 dígitos. La figura 12-2 muestra el formato del número de BCD utilizado con el coprocesador aritmético. Observe que tanto los números positivos como negativos se guardan en forma real y nunca en la forma de un complemento a diez. El directivo DT guarda los datos BCD en la memoria.

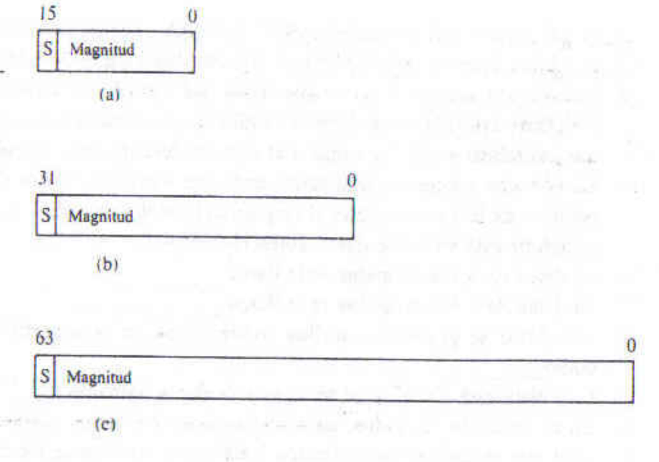


FIGURA 12–1 Formas de datos con números enteros para la familia 8087 de coprocesadores numéricos. (a) Palabra, (b) corto, y (c) largo.

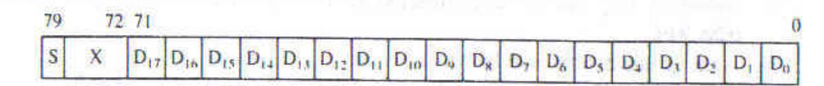


FIGURA 12–2 Formato de datos BCD para la familia 8087 de coprocesadores numéricos.

Punto decimal flotante

Los números con punto decimal flotante frecuentemente se denominan *números reales* porque son números enteros, fracciones o números mixtos. Un número con punto decimal flotante tiene tres partes: un *bit de signo*, un *exponente polarizado*, y una *mantisa*. Los números de punto flotante están escritos en *notación científica binaria*. La familia Intel de coprocesadores aritméticos soporta tres tipos de números con punto decimal flotante: corto (32 bits), largo (64 bits) y temporal (80 bits). Véase la figura 12-3 para ver las tres formas del número con el punto decimal flotante. Observe por favor que también le llamamos a la forma corta número de *precisión sencilla* y a la forma larga número de *doble precisión*. A la forma temporal de 80 bits a veces se le llama número de *precisión extendida*. Los números con punto decimal flotante, y las operaciones realizadas por el coprocesador aritmético, se apegan a la norma IEEE-754 adoptada por todos los fabricantes principales de programación. Esto incluye a Microsoft, que recientemente dejó de apoyar su formato de punto decimal flotante.

Conversión a la forma de punto decimal flotante. Convertir de la forma decimal a la forma de punto decimal flotante es una tarea sencilla que se realiza con los siguientes pasos:

1. Convierta el número decimal a binario.
2. Normalice el número binario.

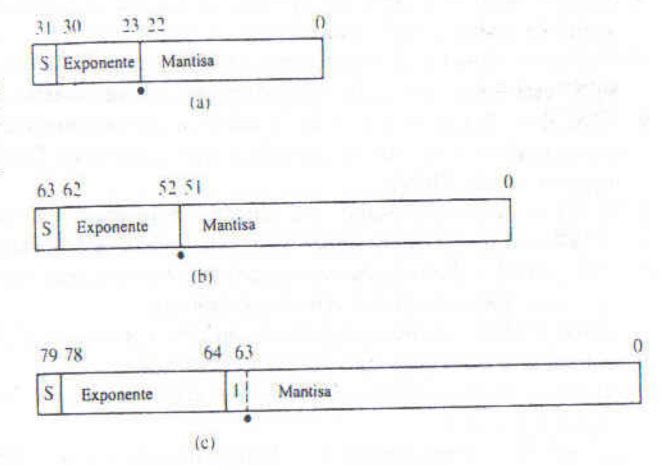


FIGURA 12–3 Números con punto decimal flotante (real). (a) Corto con una polarización de 127 (7FH) y un 1 implícito, (b) largo con una polarización de 1023 (3FFH) y un 1 implícito, y (c) temporal con una polarización de 16,383 (3FFFH) y un 1 no implícito.

3. Calcule el exponente polarizado.
4. Guarde el número en forma de punto decimal flotante.

En el ejemplo 12-2 se ilustran estos cuatro pasos para el número decimal 100.25. En este ejemplo el número decimal se convierte a un número con punto decimal flotante de precisión sencilla.

EJEMPLO 12–2

Paso	Resultado
1	100.25 = 1100100.01
2	1100100.01 = 1.10010001 × 2^6
3	110 = 110 + 01111111 = 10000101
	S Exponente Mantisa
4	0 10000101 10010001000000000000000

En el paso número 3, el exponente polarizado es el mismo exponente, un +6 (110), más una polarización de 01111111 (7FH). Todos los números de precisión sencilla utilizan una polarización de 7FH, los números de doble precisión utilizan una polarización de 3FFH, y los números de precisión extendida utilizan una polarización de 3FFFH.

El paso número 4 es donde la información se combina para generar el número con punto decimal flotante. El bit más a la izquierda es el bit de signo para el número. En este caso es un 0 porque el número es +100.25. El exponente polarizado sigue al bit del signo. La mantisa es un número de 23 bits con un bit implícito. Observe que el significante de un número 1.XXXX es la porción XXXX. El 1. es un bit implícito que solamente está guardado en la forma de precisión extendida del número con punto decimal flotante como un bit implícito.

Se aplican algunas reglas especiales a unos cuantos números. Por ejemplo, el número 0 se guarda como sólo ceros excepto por el bit del signo, que puede ser un 1 lógico para representar un cero negativo. Se guardan también la más y menos infinito como unos lógicos (1) en el exponente con una mantisa de sólo ceros y el bit de signo que representa más o menos. Un NAN (no un número) es resultado de un punto decimal flotante no válido que tiene sólo unos (1) en el exponente con una mantisa que *no* es sólo ceros.

Conversión del formato de punto flotante. La conversión de un número con punto flotante a un número decimal se resume en los siguientes pasos:

1. Separe el bit de signo, el exponente polarizado y la mantisa.
2. Convierta el exponente polarizado a un exponente real restando la polarización.
3. Escriba el número como un número binario normalizado.
4. Conviértalo a número binario no normalizado.
5. Convierta el número binario no normalizado a uno decimal.

En el ejemplo 12-3, estos cinco pasos convierten un número con punto decimal flotante de precisión sencilla a uno decimal. Observe cómo el bit de signo 1 hace que el número decimal resulte negativo. También observe que el bit 1 implícito se agrega al resultado binario normalizado en el paso número 3.

EJEMPLO 12–3

```
Paso            Resultado
1         1 100000011 10010010000000000000000

2         10000011 = 1000011 - 01111111 = 100

3         1.1001001 × 2⁴

4         11001.001

5         -25.125
```

Cómo guardar en la memoria los datos de punto decimal flotante. Los números con punto decimal flotante se guardan con el ensamblador utilizando el directivo DD para precisión sencilla, DQ para doble precisión, y DT para precisión extendida. Se muestran algunos ejemplos de almacenamiento de datos de punto flotante en el ejemplo 12-4. El ensamblador de la versión 6.0 de Microsoft contiene un error que no permite que el signo más se utilice con números positivos de punto flotante. Un +92.45 se tiene que definir como 92.45 para que el ensamblador funcione correctamente.

EJEMPLO 12–4

```
0000 C377999A             DATO7   DD   -247.6     ;define precisión sencilla
0004 40000000             DATO8   DD   2.0        ;define precisión sencilla
0008 486F4200             DATO9   DD   2.45E+5    ;define precisión sencilla
000C 4059100000000000     DATO10  DQ   100.25     ;define doble precisión
0014 3F543BF727136A40     DATO11  DQ   0.001235   ;define doble precisión
001C 400487F34D6A161E4F76 DATO12  DT   33.9876    ;define precisión extendida
```

12–2 LA ARQUITECTURA DE 80X87

El 80X87 está diseñado para operar conjuntamente con el microprocesador. Observe que el microprocesador 80486 contiene su propia versión *interna* del 80387 que es totalmente compatible. En otros miembros de la familia, el coprocesador es un circuito integrado externo que contiene la mayoría de las conexiones del microprocesador. El 80X87 ejecutará 68 instrucciones diferentes con el microprocesador. El microprocesador ejecutará todas las instrucciones normales y el 80X87 ejecutará las instrucciones del coprocesador aritmético. Para ilustrar uno de los coprocesadores, la figura 12-4 muestra el diagrama de base del coprocesador aritmético 80287, llamado algunas veces un *coprocesador numérico*.

Definiciones de las terminales del 80287

La siguiente lista describe todas las terminales del coprocesador aritmético 80287:

1. CLK — Entrada de reloj: proporciona al 80287 la señal de temporización básica.
2. CLM — Modo de reloj: selecciona si la entrada de CLK se divide por 3 o se utiliza directamente. Un 1 lógico en esta terminal hará que la entrada de CLK se utilice como la señal interna del reloj.
3. CLK286 — Entrada de reloj del 80286: en la mayoría de los sistemas está conectado la terminal CLK del 80286.
4. RESET — Entrada de reinicialización: se utiliza para inicializar el 80287.
5. D15-DO — Canal de Datos: utilizado para capturar datos y comandos del canal de datos del sistema así como para transferir información al canal de datos del sistema.
6. $\overline{\text{BUSY}}$ — Salida de ocupado: una señal que indica que el 80287 está ocupado ejecutando una instrucción. Esta terminal está conectada a la terminal BUSY del microprocesador 80286. TEST en el 8086/8088 es igual que la terminal de BUSY.

FIGURA 12-4 El coprocesador aritmético 80827.

7. $\overline{\text{ERROR}}$ — Salida de error: una señal que indica una condición de error no enmascarado y es un reflejo directo del bit de estado ES.
8. PEREQ — Solicitud de transferencia de datos de operando del coprocesador: indica que el 80287 está listo para transferir datos por medio de su conexión al canal de datos.
9. $\overline{\text{PEACK}}$ — Reconocimiento de la solicitud de transferencia de datos de operando del coprocesador: una entrada que indica que la señal de PEREQ fue reconocida por el microprocesador 80286.
10. $\overline{\text{NPRD}}$ — Leer coprocesador: esta entrada permite una transferencia de datos del 80287.
11. $\overline{\text{NPWR}}$ — Escribir en el coprocesador: esta entrada escribe datos al 80287.
12. $\overline{\text{NPS1}}$, NPS2 — Seleccionar coprocesador: estas líneas seleccionan al 80287 para que pueda realizar operaciones de coprocesador aritmético.
13. CMD1, CMD0 — Líneas de comando: dirigen la operación del 80287. Estas terminales normalmente se conectan a A2 y A1 respectivamente.
14. HLDA — Reconocimiento de cesión del canal: se conecta directamente a la terminal de HLDA del 80286.
15. COD/$\overline{\text{INTA}}$ — Reconocimiento de Código/Interrupción: se conecta a la misma terminal del 80286.
16. Vcc — Suministro de Energía: está conectado al canal de alimentación de +5.0 V del sistema.
17. Vss — Tierra: está conectado a la tierra del sistema.

Estructura interna del 80287

La figura 12-5 muestra la estructura interna del coprocesador aritmético. Observe que este dispositivo se divide en dos secciones principales: la *unidad de control* y la *unidad de ejecución numérica.*

La unidad de control conecta el coprocesador al canal de datos del sistema del microprocesador. Ambos dispositivos monitorean el flujo de instrucciones. Si la instrucción es de ESC (coprocesador), la ejecutará el coprocesador, y si no lo es la ejecutará el microprocesador.

La unidad de ejecución numérica (NEU) es responsable de ejecutar todas las 68 instrucciones. La NEU tiene una pila de ocho registros que contiene los operandos para las instrucciones aritméticas y para los resultados de instrucciones aritméticas. Las instrucciones direccionan los datos en registros de datos específicos de la pila o utilizan un mecanismo de salvar y recuperar para guardar y sacar datos. Otros registros de la NEU son el de estado, control, marcador y apuntadores de excepción.

La pila dentro del coprocesador contiene ocho registros que son de 80 bits de ancho cada uno. Estos registros de pila siempre contienen un número con punto decimal flotante de precisión extendida de 80 bits. La única vez que los datos aparecen en cualquier otra forma es cuando se transfieren entre el coprocesador y el sistema de la memoria.

Registro de estado. El registro de estado (véase la figura 12-6) refleja la operación general del coprocesador. Se accesa el registro de estado ejecutando la instrucción (FSTSW) que guarda el contenido de éste en una palabra de la memoria. La instrucción de FSTSWAX copia el registro de estado directamente al registro de AX. Una vez que el estado se guarda en la memoria o en el AX, las posiciones de bit del registro de estado pueden ser examinadas con programación normal.

A continuación se encuentra una lista de los bits de estado y su aplicación:

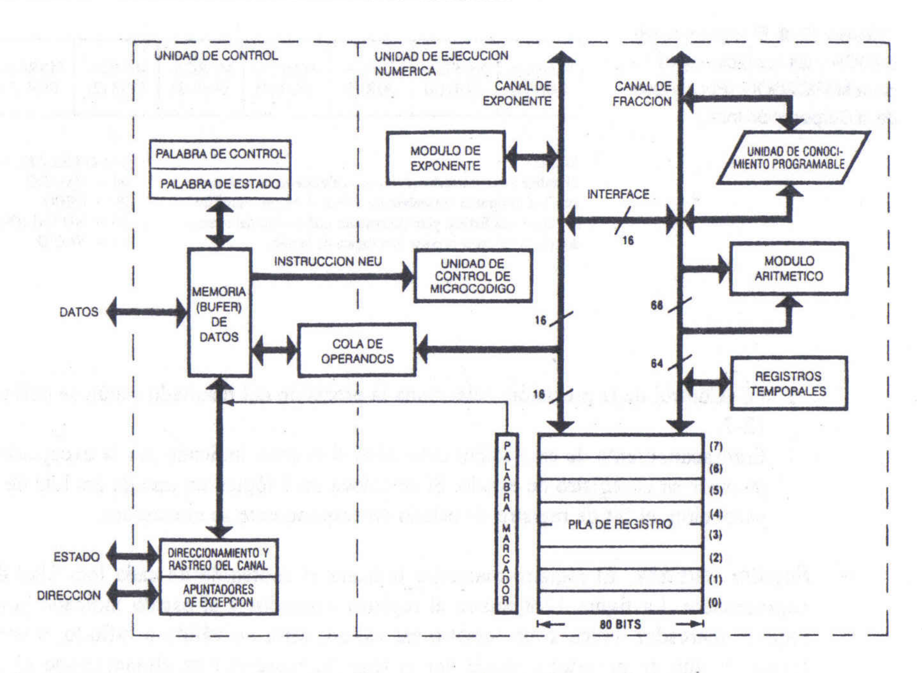


FIGURA 12–5 La estructura interna del 80X87. (Por cortesía de la Corporación Intel.)

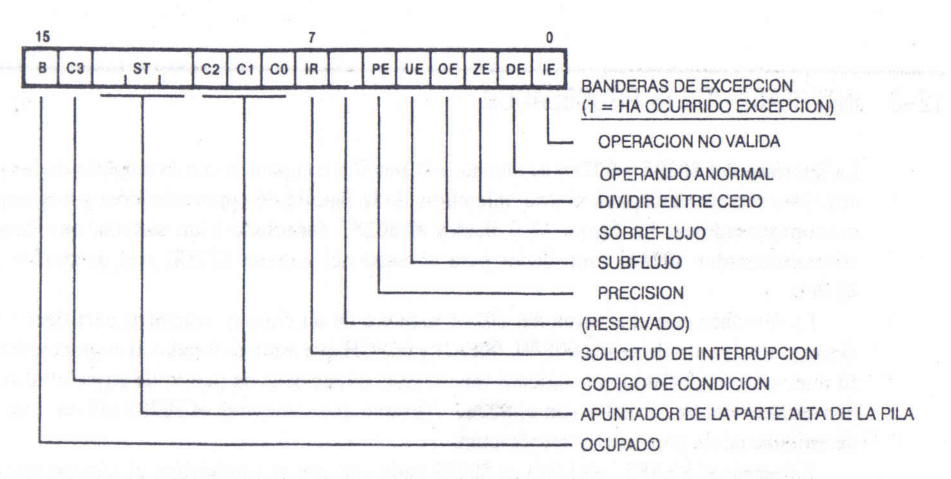


FIGURA 12–6 Palabra de estado de la familia 80X87. (Por cortesía de la Corporación Intel.)

1. B — Ocupado: indica que el coprocesador está ocupado ejecutando una tarea.
2. C3-CO — Bits para el Código de la Condición: véase la tabla 12-1 para una lista completa de cada combinación de estos bits y su función. Observe que estos bits tienen distintos significados para las diferentes instrucciones.
3. TOP — Parte alta de la pila: indica el registro actual direccionado como la parte alta de la pila.
4. ES — Resumen de errores: se activa si está activo algún bit (PE, UE; OE; ZE; DE o IE) de error sin enmascarar.
5. PE — Error de precisión: si el resultado o los operandos exceden la precisión seleccionada.
6. UE — Error de subflujo: indica un resultado diferente de cero que es demasiado pequeño para representarlo.
7. OE — Error de sobreflujo: indica un resultado que es demasiado grande para ser representado. Si este error se enmascara, el coprocesador genera infinito.
8. ZE — Error Cero: indica que el divisor era cero mientras que el dividendo es un número no infinito, no cero.
9. DE — Error fuera de lo normal: indica que por lo menos uno de los operantes no está normalizado.

TABLA 12-1 Los bits del código de condición para el registro de estado del 80287

Instrucción	*C3*	*C2*	*C1*	*C0*	*Función*
FTST,FCOM	0	0	X	0	ST > Fuente o (0 FTST)
	0	0	X	1	ST < Fuente o (0 FTST)
	1	0	X	0	ST = Fuente o (0 FTST)
	1	1	X	1	ST no es comparable
FPREM	Q1	0	Q0	Q2	3 bits más a la derecha del cociente
	?	1	?	?	incompleto
FXAM	0	0	0	0	+ subanormal
	0	0	0	1	+ NAN
	0	0	1	0	- subanormal
	0	0	1	1	+ NAN
	0	1	0	0	+ normal
	0	1	0	1	+ ∞
	0	1	1	0	- normal
	0	1	1	1	- ∞
	1	0	0	0	+ 0
	1	0	0	1	vacío
	1	0	1	0	- 0
	1	0	1	1	vacío
	1	1	0	0	+ anormal
	1	1	0	1	vacío
	1	1	1	0	- anormal
	1	1	1	1	vacío

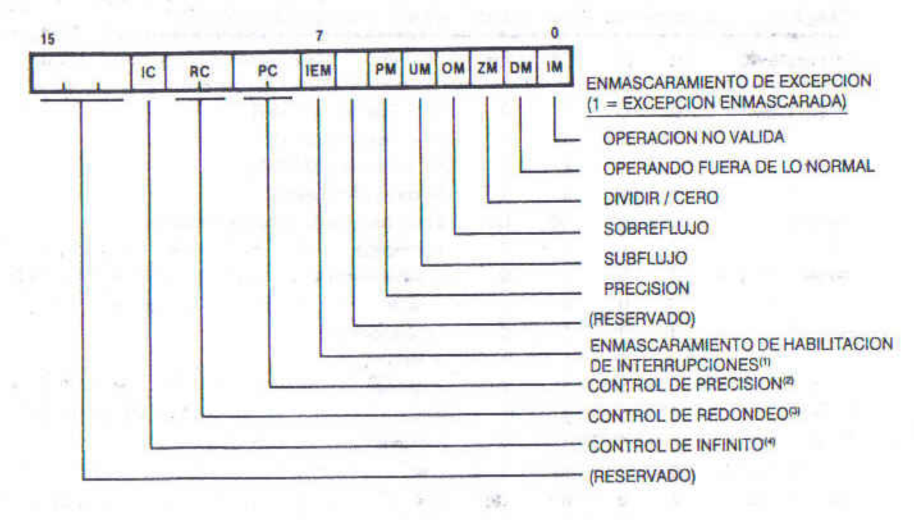
Notas: subanormal = bits iniciales de la mantisa son cero, anormal = el exponente en su valor más negativo, normal = forma estándar de punto decimal flotante, y NAN (no es número) = un exponente de unos y una mantisa que no es igual a cero.

10. IE — Error de no válido: indica un sobreflujo o subflujo de pila, forma indeterminada (0/0, ∞, −∞, etcétera) o el uso de NAN como operando. Esta bandera indica los errores como aquellos producidos al tomar la raíz cuadrada de un número negativo, etcétera.

Registro de control. El registro de control está representado en la figura 12-7. La palabra de control seleccionará precisión, control del redondeo, y control del infinito. También enmascara y desenmascara los bits de excepción que corresponden a los 6 bits más a la derecha del registro de estado. La instrucción de FLDCW se utiliza para cargar un valor en el registro de control.

Enseguida se encuentra una descripción de cada bit o agrupación de bits encontrados en el registro de control:

1. IC–Control del infinito: selecciona al infinito como afinidad o proyectivo. En afinidad permite un infinito positivo y negativo, mientras que en proyectivo supone que el infinito es sin signo.
2. RC–Control del redondeo: selecciona el tipo de redondeo según se define en la figura 12-7.



(1) Enmascaramiento de habilitación de interrupciones:
0 = Interrupciones habilitadas
1 = Interrupciones deshabilitadas (enmascaradas)

(2) Control de precisión:
00 = 24 bits
01 = (reservado)
10 = 53 bits
11 = 64 bits

(3) Control de redondeo:
00 = Redondear al más cercano o par
01 = Redondear hacia abajo (hacia −∞)
10 = Redondear hacia arriba (hacia +∞)
11 = Recortar (truncar hacia cero)

(4) Control de infinito:
0 = Proyectivo
1 = Afinidad

FIGURA 12-7 Registro de control de la familia 80X87. (Por cortesía de la Corporación Intel.)

FIGURA 12-8 El registro MARCADOR y las condiciones de cada MARCADOR. (Por cortesía de la Corporación Intel.)

15							0
MARCA-DOR (7)	MARCA-DOR (6)	MARCA-DOR (5)	MARCA-DOR (4)	MARCA-DOR (3)	MARCA-DOR (2)	MARCA-DOR (1)	MARCA-DOR (0)

NOTA:
El índice i del marcador (i) no está relacionada con la parte alta. Un programa normalmente utiliza el campo "alto" de la Palabra de Estado para determinar qué campo del marcador (i) se refiere a la parte alta lógica de la pila.

VALORES DEL MARCADOR:
00 = VALIDO
01 = CERO
10 = NO VALIDO o INFINITO
11 = VACIO

3. PC–Control de la precisión: selecciona la precisión del resultado según se define en la figura 12-7.
4. Enmascaramiento de excepción: determina si el error indicado por la excepción afecta el bit de error en el registro de estado. Si se coloca un 1 lógico en uno de los bits de control de la excepción, el bit de registro de estado correspondiente se enmascara.

Registro marcador. El registro marcador indicará el contenido de cada localidad de la pila del coprocesador. La figura 12-8 ilustra el registro marcador y el estado indicado por cada bit. El registro marcador indica si un registro es: válido, cero, no válido o infinito, o vacío. La única forma en que un programa puede ver el registro marcador es almacenando el ambiente del coprocesador utilizando las instrucciones de FSTENV, FSAVE o FRSTOR. Cada una de estas instrucciones guardará el registro marcador junto con otros datos del coprocesador.

12-3 INTERFACE CON EL PROCESADOR

La interface del 80287 y 80286 es mucho más sencilla comparada con la mayoría de los periféricos. Esto también se aplica a otros miembros de la familia de coprocesadores y sus respectivos microprocesadores. La figura 12-9 ilustra el 80287 conectado a un sistema que contiene el microprocesador 80286, controlador para el canal del sistema 82288, y el generador de reloj 82284.

La interface con el sistema del 80286 requiere de un circuito adicional para decodificar las direcciones de entrada/salida 00F8H, 00FAH y 00FCH que están dedicadas al coprocesador 80287. El microprocesador automáticamente genera estas direcciones de puerto de entrada/salida durante comunicaciones normales con el 80287. Observe que solamente el 80287 utiliza estos puertos de entrada/salida para esta comunicación.

La terminal RESET inicializa el 80287 cada vez que se reinicializa el microprocesador. El 80287 responde a una entrada de reinicialización o a la instrucción de reinicialización por programa, FINIT, de casi la misma forma. El circuito de reinicialización obliga al 80287 a operar en modo real, y la instrucción de FINIT no cambia el modo. La tabla 12-2 muestra cómo las distintas secciones internas del 80287 responden a la reinicialización. En todos, menos los casos especiales, se opera el 80287 en el modo de operación de reinicialización.

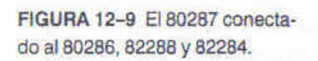


FIGURA 12–9 El 80287 conectado al 80286, 82288 y 82284.

12–4 EL CONJUNTO DE INSTRUCCIONES

El coprocesador aritmético ejecuta 68 instrucciones distintas. Cada vez que una instrucción del coprocesador hace referencia a la memoria, el microprocesador automáticamente genera la dirección en la memoria de esa instrucción. El coprocesador utiliza el canal de datos para la transferencia de datos durante las instrucciones del coprocesador y el microprocesador lo utiliza durante las instrucciones normales.

TABLA 12–2 Estado interno del 80287 después de una REINICIALIZACION o una instrucción FINIT

Campo	*Valor*	*Condición*
Infinito	0	Proyectivo
Redondeo	00	Redondear al más cercano
Precisión	11	64 bits
Sin definir errores	11111	Bits de error desactivado
Ocupado	0	No ocupado
C3-C0	????	Desconocido
ALTO	000	Registro 000
ES	0	No error
Bits de error	00000	No errores
Marcadores	11	Vacío
Registros	—	No cambiado

Esta sección del texto describe la función de cada instrucción e indica su forma en lenguaje ensamblador. Ya que el coprocesador utiliza los modos de direccionamiento de memoria del microprocesador, no se ilustran todas las formas posibles de cada instrucción. Cada vez que el ensamblador encuentra uno de los mnemónicos de código de instrucción del coprocesador, lo convierte en una instrucción de ESC en el lenguaje de máquina. La instrucción ESC representa un código de instrucción para el coprocesador.

Instrucciones para la transferencia de datos

Existen tres transferencias de datos básicas: punto flotante, número entero con signo y BCD. La única vez que los datos aparecen en la forma de número entero con signo o en BCD es en la memoria. Dentro del coprocesador, siempre se guardan los datos como un número de punto flotante de precisión extendida de 80 bits.

Transferencia de datos de punto flotante. Hay cuatro instrucciones para la transferencia de datos de punto flotante en el conjunto de instrucciones del coprocesador: FLD (carga real), FST (guardar real), FSTP (guardar real y recuperar real) y FXCH (intercambiar).

La instrucción de FLD carga los datos de la memoria en la parte alta de la pila interna. Esta instrucción guarda los datos en la parte más alta de la pila y luego decrementa el apuntador de pila uno. Los datos cargados en la pila son de cualquier localidad de la memoria o de otro registro del coprocesador. Por ejemplo, una instrucción FLD ST(2) copia el contenido del registro 2 a la parte alta de la pila, que es ST. La parte alta de la pila es el registro 0 cuando se inicializa o se reinicializa el coprocesador. Otro ejemplo es la instrucción de FLD DATO7, que copia el contenido de la localidad de la memoria DATO7 en la parte superior de la pila. El tamaño de la transferencia lo determina automáticamente el ensamblador por medio de los directivos DD para la precisión sencilla, DQ para la doble precisión, y DT para la precisión extendida.

La instrucción de FST almacena una copia de la parte alta de la pila en la localidad de la memoria o en el registro de coprocesador indicado por el operando. Al momento de almacenar, el número interno de punto flotante de precisión extendida se redondea al tamaño del número del punto decimal flotante indicado por el registro de control.

La instrucción de FSTP (guardar y recuperar punto flotante) guarda una copia de la parte alta de la pila en la memoria o en cualquier registro de coprocesador y luego recupera los datos de la parte alta de la pila. Se podría considerar la FST como una instrucción de *copiado* y la FSTP como una instrucción de *remover*.

La instrucción de FXCH intercambia el registro indicado por el operando con la parte alta de la pila. Por ejemplo, la instrucción de FXCH ST(2) intercambia la parte alta de la pila con el registro 2.

Instrucciones para la transferencia de datos de números enteros. El coprocesador soporta tres instrucciones de transferencia de datos de números enteros: FILD (cargar número entero), FIST (guardar número entero) y FISTP (guardar y recuperar número entero). Estas tres instrucciones funcionan como lo hacían FLD, FST y FSTP, excepto que los datos transferidos son datos de números enteros. El coprocesador convierte automáticamente los datos internos de precisión extendida a datos de números enteros. El tamaño de los datos se determina por la forma en que la etiqueta se define con DW, DD o DQ.

Instrucciones de transferencia de datos en BCD. Dos instrucciones cargan o almacenan datos en BCD. La instrucción de FBLD carga la parte alta de la pila con datos de memoria en BCD y el FBSTP guarda la parte alta de la pila y hace una recuperación.

EJEMPLO 12–5

```
                                  .286
                                  .287

0000                              DATOS   SEGMENT

0000 41F00000                     DATO1   DD    30.0        ;precisión sencilla
0004 0000000000003E40             DATO2   DQ    30.0        ;precisión doble
000C 00000000000000F00340         DATO3   DT    30.0        ;precisión extendida

0016 001E                         DATO4   DW    30          ;número entero de 16 bits
0018 0000001E                     DATO5   DD    30          ;número entero de 32 bits
001C 1E00000000000000             DATO6   DQ    30          ;número entero de 64 bits

0024 30000000000000000000         DATO7   DT    30H         ;BCD 30

002E                              DATOS   ENDS

0000                              CODIGO  SEGMENT

                                          ASSUME CS:CODE,DS:DATAS

0000 D9 06 0000 R                         FLD  DATO1
0004 DD 06 0004 R                         FLD  DATO2
0008 DB 2E 000C R                         FLD  DATO3
000C DF 06 0016 R                         FILD DATO4
0010 DB 06 0018 R                         FILD DATO5
0014 DF 2E 001C R                         FILD DATO6

0018 DF 26 0024 R                         FBLD DATO7

001C                              CODIGO  ENDS
                                          END
```

El ejemplo 12-5 muestra cómo el ensamblador ajusta automáticamente las instrucciones de FLD, FILD y FBLD para operandos de distintos tamaños. (Examine cuidadosamente los códigos de máquina de las instrucciones.) Observe que este ejemplo empieza con los directivos .286 y .287 que identifican el microprocesador como un 80286 y el coprocesador como un 80287. Deberá tomar en cuenta que si el microprocesador 80386 está en uso con su coprocesador, aparecen las directivas .386 y .387. El ensamblador asume en forma implícita que el programa está ensamblado para un 8086/8088 con un coprocesador 8087.

Instrucciones aritméticas

Las instrucciones aritméticas para el coprocesador incluyen suma, resta, multiplicación, división y raíz cuadrada. Las instrucciones relacionadas con aritmética son escalamiento, redondeo, valor absoluto y cambios de signo.

La tabla 12-3 muestra los modos básicos para direccionamiento permitidos para las operaciones aritméticas. Cada modo de direccionamiento se muestra con un ejemplo utilizando la instrucción FADD (suma real). Todas las operaciones aritméticas son en punto decimal flotante excepto en algunos casos cuando datos de la memoria son referidos como un operando.

La forma clásica de una pila para direccionar los datos de un operando (direccionamiento de pila) utiliza la parte alta de la pila como el operando fuente y la localidad siguiente de la pila como el operando destino. Posteriormente, los dos datos originales se retiran de la pila y sólo queda el resultado en la parte alta de la pila. (Internamente el coprocesador recupera la fuente de la parte alta de la pila). Para utilizar este modo de direccionamiento, la instrucción se coloca en el programa sin ningún operando como FADD o FSUB. La instrucción de FADD suma el contenido de la parte alta de la pila (ST) al contenido de la localidad siguiente de la pila (ST(1)) y guarda la respuesta en la parte alta de la pila, también quita de la pila los dos datos originales recuperándolos.

El modo de direccionamiento por registro utiliza ST para la parte alta de la pila y ST(n) para otra localidad donde n es el número del registro. Con esta forma, un operando debe estar en ST y el otro en ST(n). Observe que para duplicar la parte alta de la pila se utiliza la instrucción de FADD ST,ST (0) donde ST(0) direcciona la parte alta de la pila. Uno de los dos operandos en el modo de direccionamiento por registro debe ser ST, mientras que el otro debe estar en la forma ST(n), donde n es un registro de pila 0-7.

El modo de direccionamiento en memoria siempre utiliza como destino la parte alta de la pila porque el coprocesador es una máquina orientada a pilas. Por ejemplo, la instrucción de FADD

TABLA 12-3 Modos de direccionamiento aritméticos

Modo	*Forma*	*Ejemplos*
Pila	ST,ST(1)	FADD
Registro	ST,SAT(n)	FADD ST,ST(2)
	ST(n),ST	FADD ST(6),ST
Recuperar registro	ST(n),ST	FADD ST(3),ST
Memoria	operando	FADD DATOS

Nota: El direccionamiento de la pila se fija como ST,ST(1) y n = número de registro 0-7.

DATA suma el número real del contenido de los datos de una localidad de la memoria a la parte alta de la pila.

Operaciones aritméticas. La letra P en un código de instrucción especifica la recuperación de un registro de la pila después de la operación (FADDP comparada con FADD). La letra R en un código de instrucción (sólo resta y división) indica modo inverso. El modo inverso es útil para los datos de memoria porque normalmente éstos se restan del contenido de la parte alta de la pila. Una instrucción de resta inversa resta de la parte alta de la pila el contenido de una localidad memoria y guarda el resultado en la parte alta de la pila. Por ejemplo, si la parte alta de la pila contiene un 10 y la localidad de memoria DATO1 contiene un 1, entonces la instrucción de FSUB DATA1 resulta en un +9 en la parte alta de la pila y la instrucción de FSUBR resulta como –9.

La letra I como segunda letra en un código de instrucción indica que el operando de memoria es un número entero. Por ejemplo, la instrucción de FADD DATO es una suma en punto flotante, mientras que FIADD DATO es una suma enteros que suma el número entero de la localidad de memoria DATO al número de punto flotante en la parte alta de la pila. Se aplican las mismas reglas a las instrucciones FADD, FSUB, FMUL y FDIV.

Operaciones relacionadas con aritmética. Otras operaciones que son de naturaleza aritmética incluyen: FSQRT (raíz cuadrada), FSCALE (escalar un número), FPREM (encontrar residuo parcial), FRNDINT (redondear a un número entero), FXTRACT (extraer exponente y mantisa), FABS (encontrar valor absoluto) y FCHG (cambiar signo). Siguen estas instrucciones y las funciones que realizan:

1. FSQRT — encuentra la raíz cuadrada de la parte alta de la pila y deja en la parte alta de la pila la raíz cuadrada que resulta. Ocurre un error de no válido para la raíz cuadrada de un número negativo. Por esta razón, el bit de IE del registro de estado se deberá probar cada vez que pueda ocurrir un resultado no válido.
2. FSCALE — suma el contenido de ST(1) (interpretado como número entero) al exponente de la parte alta de la pila. FSCALE puede multiplicar o dividir rápidamente por una potencia de dos. El valor de ST(1) deberá estar entre 2^{-15} y 2^{+15}.
3. FPREM — realiza la división del módulo de ST por ST(1). El residuo que resulta se encuentra en la parte alta de la pila y tiene el mismo signo que el dividendo original.
4. FRNDINT — redondea la parte alta de la pila a un número entero.
5. FXTRACT — divide el número de encima de la pila en dos números separados que representan el valor del exponente y el valor de la mantisa. La mantisa extraído se encuentra en la parte alta de la pila y el exponente en ST(1). Frecuentemente se utiliza esta instrucción para convertir un número con punto decimal flotante a una forma que se pueda imprimir.
6. FABS — cambia a positivo el signo de la parte alta de la pila.
7. FCHS — cambia el signo de positivo a negativo o viceversa.

Instrucciones de comparación

Todas las instrucciones de comparación examinan datos de la parte alta de la pila en relación con otro elemento y devuelven el resultado de la comparación en los bits C3–CO para el código de la condición del registro de estado. Las comparaciones que permite el coprocesador son: FCOM (comparación de punto flotante), FCOMP (comparación de punto decimal flotante con una recuperación), FCOMPP (comparación de punto decimal flotante con 2 recuperaciones), FICOM (com-

paración de número entero), FICOMP (comparación de número entero y recuperación), FSTS (probar) y FXAM (examinar). A continuación se encuentra una lista de estas instrucciones con una descripción de su función:

1. FCOM — compara el dato de punto flotante de la parte alta de la pila con un operando, que puede ser cualquier registro o cualquier operando de memoria. Si el operando no está codificado con la instrucción, se compara el siguiente elemento de la pila ST(1) con la parte alta de la pila ST.
2. FCOMP y FCOMPP — ambas instrucciones funcionan como FCOM, pero también recupera uno o dos registros de la pila.
3. FICOM y FICOMP — la parte alta de la pila se compara con el número entero almacenado en un operando de memoria. Además de la comparación, FICOMP también recupera la parte alta de la pila.
4. FTST — prueba el contenido de la parte alta de la pila contra cero. El resultado de la comparación se codifica en los bits para el código de la condición del registro de estado como se ilustra en la tabla 12-1 con el registro de estado.
5. FXAM — examina la parte alta de la pila y modifica los bits para el código de la condición, para indicar si el contenido es positivo, negativo, normalizado, etcétera. Vea nuevamente el registro de estado en la tabla 12-1.

Operaciones trascendentales

Las instrucciones trascendentales incluyen: FPTAN (tangente parcial), FPATAM (tangente de arco parcial), F2XM1 ($2^x - 1$), FYL2X (Y $\log_2$ X), y FYL2XP1 (Y Log_2(X+1)). Sigue a continuación una lista de estas operaciones con una descripción de cada operación trascendental:

1. FPTAN — encuentra el tangente parcial de Y/X = tan θ. El valor de θ está en la parte alta de la pila y debe estar entre 0 y π /4. El resultado es un cociente encontrado como ST = X y ST(1) = Y. Si el valor está fuera del campo permisible, ocurre un error de no válido como lo indica el registro de estado.
2. FPATAN — encuentra el tangente parcial como θ = ARCTAN X/Y. El valor de X está en la parte alta de la pila y Y está en ST(1). Los valores de X y Y deben de ser como sigue: $O < Y < X < \infty$. La instrucción recupera la parte alta de la pila y deja a θ ahí.
3. F2AM1 — encuentra la función $2^x - 1$. El valor de X se toma de la parte alta de la pila y el resultado regresa a la parte alta de la pila. Para obtener 2^x, suma uno al resultado de la parte alta de la pila. Esta función se puede utilizar para derivar las funciones indicadas en la tabla 12-4. Observe que las constantes de $\log_2{}^{10}$ y $\log_2{}^{e}$ están integrados como valores normales para el coprocesador.
4. FYL2X — encuentra Y $\log_2$X. El valor X se toma de la parte alta de la pila y Y se toma de ST(1). El resultado se encuentra en la parte alta de la pila después de una recuperación. El valor de X debe variar entre 0 y ∞ el valor de Y debe estar entre $-\infty$ y $+ \infty$.
5. FYL2XPI — encuentra Y $\log_2$(X + 1). El valor de X se toma de la parte alta de la pila y Y se toma de ST(1). El resultado se encuentra en la parte alta de la pila después de una recuperación. El valor de X debe variar entre 0 y $1- \sqrt{2}/2$ y el valor de Y debe variar entre $-\infty$ y $+ \infty$.

TABLA 12-4 Funciones exponenciales

Función	*Ecuación*
10^x	$2^x \log 2^{10}$
e^x	$2^x \log 2^e$
Y^x	$2^x \log 2^Y$

Operaciones con constantes

El conjunto de instrucciones del coprocesador incluye códigos de instrucción que regresan las constantes a la parte alta de la pila. En la tabla 12-5 aparece una lista de estas instrucciones.

Instrucciones de control del coprocesador

El coprocesador tiene instrucciones de control para inicialización, manejo de excepciones, e intercambio de tareas. Las instrucciones de control tienen dos formas. Por ejemplo, FINIT inicializa el coprocesador y también lo hace FNINIT. La diferencia es que FNINIT no causa ningún estado de espera, mientras que FINIT sí causa esperas. El microprocesador espera la instrucción de FINIT probando la terminal BUSY del coprocesador. Todas las instrucciones de control tienen estas dos formas. A continuación se encuentra una lista de cada instrucción de control con su función:

1. FINIT/FNINIT—esta instrucción realiza la misma función básica que la reinicialización según se describe en la sección 12-3. El coprocesador opera con una aproximación de redondeo de proyección, que redondea al más cercano, y utiliza una precisión de 64 bits cuando se reinicializa o inicializa.
2. FSETPM—cambia el modo de direccionamiento del coprocesador al modo de direccionamiento protegido. Este modo se utiliza cuando el microprocesador también se opera en el modo protegido. Así como con el microprocesador, solamente se puede salir del modo protegido por medio del circuito de reinicialización o en el caso del 80386/80486 cambiándose al registro de control.
3. FLDCW—carga el registro de control con la palabra direccionada por el operando.
4. FSTCW/FNSTCW—carga el registro de control en el operando de memoria de tamaño palabra.

TABLA 12-5 Operaciones de constantes

Instrucción	*Constante transferida a ST*
FLDZ	+0.0
FLD1	+1.0
FLDPI	π
FLDL2T	$\log_2 10$
FLDL2E	$\log_2 e$
FLDLG2	$\log_{10} 2$
FLDLN2	$\log_e 2$

5. FSTSW AX/FNSTSW AX — copia el contenido del registro de control al registro AX.
6. FCLEX/FNCLEX—desactiva las banderas de error del registro de estado y también la bandera de ocupado.
7. FSAVE/FNSAVE—escribe el estado completo de la máquina en la memoria. La figura 12-10 muestra la disposición de la memoria para esta instrucción.
8. FRSTOR — Recupera el estado de la máquina de la memoria. Esta instrucción se utiliza para recuperar la información salvada por FSAVE/FNSAVE.
9. FSTENV/FNSTENV—almacena el ambiente del coprocesador como se muestra en la figura 12-11.
10. FLDENV—vuelve a cargar el ambiente guardado por FSTENV/FNSTENV.
11. FINCST—incrementa el apuntador de pila.
12. FDECSTP—decrementa el apuntador de pila.

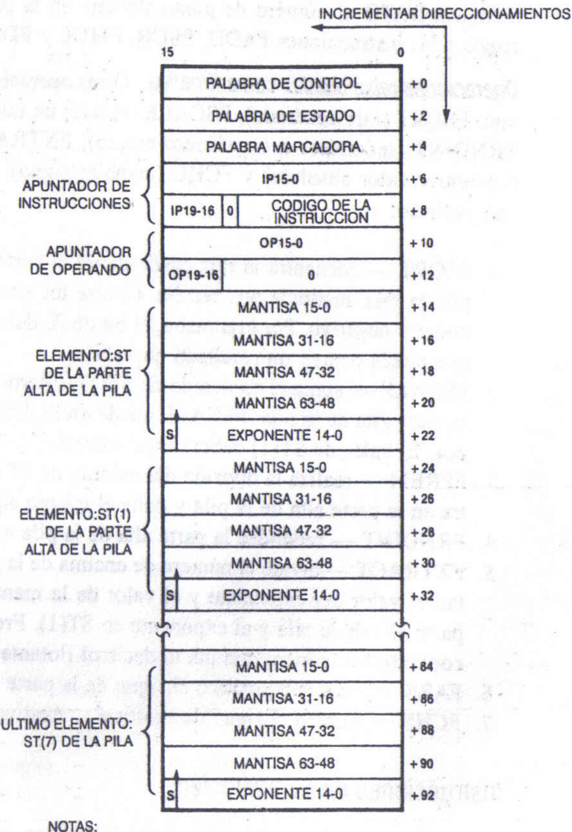


FIGURA 12-10 Formato de la memoria cuando los registros del 8087 se almacenan mediante la instrucción FSAVE. (Por cortesía de la Corporación Intel.)

13. FFREE–libera un registro cambiando el registro marcador destino a vacío. No afecta el contenido del registro.
14. FNOP–NOP del coprocesador de punto flotante.
15. FWAIT–hace que el microprocesador espere a que el coprocesador termine una operación. FWAIT se debe utilizar antes de que el microprocesador accese datos de memoria afectados por el coprocesador.

Instrucciones para el 80387/80487

Aunque todavía nos falta hablar del microprocesador 80386 y su compañero coprocesador el 80387, del 80486DX y su coprocesador integrado, o del 80486SX y de su compañero coprocesador 80487SX, podemos discutir el conjunto de instrucciones de estos coprocesadores y las diferencias que existen con las otras versiones del coprocesador. Estos coprocesadores más nuevos contienen las mismas instrucciones básicas proporcionadas por las versiones anteriores, con unas cuantas instrucciones adicionales.

El 80387, 80486 y 80486SX contienen las siguientes instrucciones adicionales: FCOS (coseno), FPREM1 (residuo parcial), FSIN (seno), FSINCOS (seno y coseno), y FUCOM/FUCOMP/FUCOMPP (una comparación desordenado). Las instrucciones de seno y coseno son los elementos más importantes que se han agregado al conjunto de instrucciones. En las versiones anteriores del coprocesador era necesario calcular seno y coseno de la tangente.

La tabla 12-6 indica los conjuntos de instrucciones para todas las versiones del coprocesador. También indica el número de periodos del reloj necesarios para ejecutar cada instrucción. Se



FIGURA 12–11 El formato de memoria para la instrucción de FSTENV. (Por cortesía de la Corporación Intel.)

TABLA 12-6 El conjunto de instrucciones del coprocesador aritmético

F2XM1 $2^{ST}-1$

11011001 11110000

Ejemplo		Ciclos de reloj
F2XM1	8087	310—630
	80287	310—630
	80387	211—476
	80486/7	140—279

FABS Valor absoluto de ST

11011001 11100001

Ejemplo		Ciclos de reloj
FABS	8087	10—17
	80287	10—17
	80387	22
	80486/7	3

FADD/FADDP/FIADD Suma

11011000 oo000mmm disp	Memoria de 32 bits (FADD)
11011100 oo000mmm disp	Memoria de 64 bits (FADD)
11011d00 11000rrr	FADD ST,ST(rrr)
11011110 11000rrr	FADDP ST,ST(rrr)
11011110 oo000mmm disp	Memoria de 16 bits (FIADD)
11011010 oo000mmm disp	Memoria de 32 bits (FIADD)

Formato	Ejemplos		Ciclos de reloj
FADD FADDP FIADD	DATOS FADD FADD ST,ST(1) FADDP NUMERO FIADD FADD ST,ST(3) FADDP ST,ST(2) FADD ST(2),ST	8087	70—143
		80287	70—143
		80387	23—72
		80486/7	8—20

FCLEX/FNCLEX Borrar errores

11011011 11100010

Ejemplo		Ciclos de reloj
FCLEX FNCLEX	8087	2—8
	80287	2—8
	80387	11
	80486/7	7

TABLA 12-6 (continuación)

FCOM/FCOMP/FCOMPP/FICOM/FICOMP Comparar

11011000 oo010mmm disp	Memoria de 32 bits (FCOM)
11011100 oo010mmm disp	Memoria de 64 bits (FCOM)
11011000 11010rrr	FCOM ST(rrr)
11011000 oo011mmm disp	Memoria de 32 bits (FCOMP)
11011100 oo011mmm disp	Memoria de 64 bits (FCOMP)
11011000 11011rrr	FCOMP ST(rrr)
11011110 11011001	FCOMPP
11011110 oo010mmm disp	Memoria de 16 bits (FICOM)
11011010 oo010mmm disp	Memoria de 32 bits (FICOM)
11011110 oo011mmm disp	Memoria de 16 bits (FICOMP)
11011010 oo011mmm disp	Memoria de 32 bits (FICOMP)

Formato	Ejemplos	Ciclos de reloj	
FCOM FCOMP FCOMPP FICOM FICOMP	FCOM ST(2) FCOMP DATO FCOMPP FICOM NUMERO FICOMP DATO3	8087	40—93
		80287	40—93
		80387	24—63
		80486/7	15—20

FCOS Coseno de ST

11011001 11111111

Ejemplo	Ciclos de reloj	
FCOS	8087	—
	80287	—
	80387	123—772
	80486/7	193—279

FDECSTP Decrementa el apuntador de pila

11011001 11110110

Ejemplo	Ciclos de reloj	
FDECSTP	8087	6—12
	80287	6—12
	80387	22
	80486/7	3

FDISI/FNDISI Deshabilita interrupciones

11011011 11100001

(se ignoran en el 80287, 80387 y 80486/7)

Ejemplo		Ciclos de reloj
FDISI FNDISI	8087	2—8
	80287	—
	80387	—
	80486/7	—

FDIV/FDIVP/FIDIV División

```
11011000 oo110mmm disp    Memoria de 32 bits (FDIV)
11011100 oo100mmm disp    Memoria de 64 bits (FDIV)
11011d00 11111rrr         FDIV ST,ST(rrr)
11011110 11111rrr         FDIVP ST,ST(rrr)
11011110 oo110mmm disp    Memoria de 16 bits (FIDIV)
11011010 oo110mmm disp    Memoria de 32 bits (FIDIV)
```

Formato	Ejemplos		Ciclos de reloj
FDIV FDIVP FIDIV	FDIV DATO FDIV ST,ST(3) FDIVP FIDIV NUMERO FDIV ST,ST(5) FDIVP ST,ST(2) FDIV ST(2),ST	8087	191—243
		80287	191—243
		80387	88—140
		80486/7	8—89

FDIVR/FDIVRP/FIDIVR División inversa

```
11011000 oo111mmm disp    Memoria de 32 bits (FDIVR)
11011100 oo111mmm disp    Memoria de 64 bits (FDIVR)
11011d00 11110rrr         FDIVR ST,ST(rrr)
11011110 11110rrr         FDIVRP ST,ST(rrr)
11011110 oo111mmm disp    Memoria de 16 bits (FIDIVR)
11011010 oo111mmm disp    Memoria de 32 bits (FIDIVR)
```

Formato	Ejemplos		Ciclos de reloj
FDIVR FDIVRP FIDIVR	FDIVR DATO FDIVR ST,ST(3) FDIVRP FIDIVR NUMERO FDIVR ST,ST(5) FDIVRP ST,ST(2) FDIVR ST(2),ST	8087	191—243
		80287	191—243
		80387	88—140
		80486/7	8—89

FENI/FNENI Deshabilita interrupciones

```
11011011 11100000
```

(se ignoran en el 80287, 80387 y 80486/7)

Ejemplo		Ciclos de reloj
FENI FNENI	8087	2—8
	80287	—
	80387	—
	80486/7	—

TABLA 12-6 (continuación)

FFREE Registro libre

11011101 11000rrr

Formato	Ejemplos		Ciclos de reloj
FFREE	FFREE FFREE ST(1) FFREE ST(2)	8087	9—16
		80287	9—16
		80387	18
		80486/7	3

FINCSTP Incrementa el apuntador de pila

11011001 11110111

Ejemplo		Ciclos de reloj
FINCSTP	8087	6—12
	80287	6—12
	80387	21
	80486/7	3

FINIT/FNINIT Inicializar coprocesador

11011001 11110110

Ejemplo		Ciclos de reloj
FINIT FNINIT	8087	2—8
	80287	2—8
	80387	33
	80486/7	17

FLD/FILD/FBLD Cargar datos a ST(0)

11011001 oo000mmm disp	Memoria de 32 bits (FLD)
11011101 oo000mmm disp	Memoria de 64 bits (FLD)
11011011 oo101mmm disp	Memoria de 80 bits (FLD)
11011111 oo000mmm disp	Memoria de 16 bits (FILD)
11011011 oo000mmm disp	Memoria de 32 bits (FILD)
11011111 oo101mmm disp	Memoria de 64 bits (FILD)
11011111 oo100mmm disp	Memoria de 80 bits (FBLD)

Formato	Ejemplos		Ciclos de reloj
FLD FILD FBLD	FLD DATO FILD DATO1 FBLD DEC_DATO	8087	17—310
		80287	17—310
		80387	14—275
		80486/7	3—103

FLD1 Cargar + 1.0 a ST(0)

11011001 11101000

Ejemplo — Ciclos de reloj

Ejemplo	Procesador	Ciclos de reloj
FLD1	8087	15—21
	80287	15—21
	80387	24
	80486/7	4

FLDZ Cargar + 0.0 a ST(0)

11011001 11101110

Ejemplo — Ciclos de reloj

Ejemplo	Procesador	Ciclos de reloj
FLDZ	8087	11—17
	80287	11—17
	80387	20
	80486/7	4

FLDPI Cargar π a ST(0)

11011001 11101011

Ejemplo — Ciclos de reloj

Ejemplo	Procesador	Ciclos de reloj
FLDPI	8087	16—22
	80287	16—22
	80387	40
	80486/7	8

FLDL2E Cargar $\log_2 e$ a ST(0)

11011001 11101010

Ejemplo — Ciclos de reloj

Ejemplo	Procesador	Ciclos de reloj
FLDL2E	8087	15—21
	80287	15—21
	80387	40
	80486/7	8

TABLA 12–6 (continuación)

FLDL2T Cargar $\log_2 10$ a ST(0)

11011001 11101001

Ejemplo		Ciclos de reloj
FLDL2T	8087	16—22
	80287	16—22
	80387	40
	80486/7	8

FLDLG2 Cargar $\log_{10} 2$ a ST(0)

11011001 11101000

Ejemplo		Ciclos de reloj
FLDLG2	8087	18—24
	80287	18—24
	80387	41
	80486/7	8

FLDLN2 Cargar $\log_e 2$ a ST(0)

11011001 11101101

Ejemplo		Ciclos de reloj
FLDLN2	8087	17—23
	80287	17—23
	80387	41
	80486/7	8

FLDCW Cargar registro de control

11011001 oo101mmm disp

Formato	Ejemplos		Ciclos de reloj
FLDCW	FLDCW DATO FLDCW ESTADO	8087	7—14
		80287	7—14
		80387	19
		80486/7	4

FLDENV Cargar ambiente

11011001 oo100mmm disp

Formato	Ejemplos	Ciclos de reloj	
FLDENV	FLDENV AMBIENTE FLDENV DATO	8087	35—45
		80287	25—45
		80387	71
		80486/7	34—44

FMUL/FMULP/FIMUL Multiplicación

11011000 oo001mmm disp	Memoria de 32 bits (FMUL)
11011100 oo001mmm disp	Memoria de 64 bits (FMUL)
11011d00 11001rrr	FMUL ST,ST(rrr)
11011110 11001rrr	FMULP ST,ST(rrr)
11011110 oo001mmm disp	Memoria de 16 bits (FIMUL)
11011010 oo001mmm disp	Memoria de 32 bits (FIMUL)

Formato	Ejemplos	Ciclos de reloj	
FMUL FMULP FIMUL	FMUL DATO FMUL ST,ST(2) FMUL ST(2),ST FMULP FIMUL DATO3	8087	110—168
		80287	110—168
		80387	29—82
		80486/7	11—27

FNOP No operación

11011001 11010000

Ejemplo	Ciclos de reloj	
FNOP	8087	10—16
	80287	10—16
	80387	12
	80486/7	3

FPATAN Arco tangente parcial de ST(0)

11011001 11110011

Ejemplo	Ciclos de reloj	
FPATAN	8087	250—800
	80287	250—800
	80387	314—487
	80486/7	218—303

TABLA 12-6 (continuación)

FPREM Residuo parcial

11011001 11111000

Ejemplo	Ciclos de reloj	
FPREM	8087	15—190
	80287	15—190
	80387	74—155
	80486/7	70—138

FPREM1 Residuo parcial (IEEE)

11011001 11110101

Ejemplo	Ciclos de reloj	
FPREM1	8087	—
	80287	—
	80387	95—185
	80486/7	72—167

FPTAN Tangente parcial de ST(0)

11011001 11110010

Ejemplo	Ciclos de reloj	
FPTAN	8087	30—450
	80287	30—450
	80387	191—497
	80486/7	200—273

FRNDINT Redondear ST(0) a número entero

11011001 11111100

Ejemplo	Ciclos de reloj	
FRNDINT	8087	16—50
	80287	16—50
	80387	66—80
	80486/7	21—30

FRSTOR Recuperar estado

11011101 oo110mmm disp

Formato	Ejemplos		Ciclos de reloj
FRSTOR	FRSTOR DATO FRSTOR ESTADO FRSTOR MAQUINA	8087	197—207
		80287	197—207
		80387	308
		80486/7	120—131

FSAVE/FNSAVE Salvar estado de la máquina

11011101 oo110mmm disp

Formato	Ejemplos		Ciclos de reloj
FSAVE FNSAVE	FSAVE ESTADO FNSAVE ESTADO FSAVE MAQUINA	8087	197—207
		80287	197—207
		80387	375
		80486/7	143—154

FSCALE Escalar ST(0) por ST(1)

11011001 11111101

Ejemplo		Ciclos de reloj
FSCALE	8087	32—38
	80287	32—38
	80387	67—86
	80486/7	30—32

FSETPM Inicializar modo protegido

11011011 11100100

Ejemplo		Ciclos de reloj
FSETPM	8087	—
	80287	2—18
	80387	12
	80486/7	—

TABLA 12–6 (continuación)

FSIN Seno de ST(0)

11011001 11111110

Ejemplo		Ciclos de reloj
FSIN	8087	—
	80287	—
	80387	122—771
	80486/7	193—279

FSINCOS Encontrar seno y coseno de ST(0)

11011001 11111011

Ejemplo		Ciclos de reloj
FSINCOS	8087	—
	80287	—
	80387	194—809
	80486/7	243—329

FSQRT Raíz cuadrada de ST(0)

11011001 11111010

Ejemplo		Ciclos de reloj
FSQRT	8087	180—186
	80287	180—186
	80387	122—129
	80486/7	83—87

FST/FSTP/FIST/FISTP/FBSTP Almacenar

11011001 oo010mmm disp	Memoria de 32 bits (FST)
11011101 oo010mmm disp	Memoria de 64 bits (FST)
11011101 11010rrr	FST ST(rrr)
11011011 oo011mmm disp	Memoria de 32 bits (FSTP)
11011101 oo011mmm disp	Memoria de 64 bits (FSTP)
11011011 oo111mmm disp	Memoria de 80 bits (FSTP)
11011101 11001rrr	FSTP ST(rrr)
11011111 oo010mmm disp	Memoria de 16 bits (FIST)
11011011 oo010mmm disp	Memoria de 32 bits (FIST)
11011111 oo011mmm disp	Memoria de 16 bits (FISTP)
11011011 oo011mmm disp	Memoria de 32 bits (FISTP)
11011111 oo111mmm disp	Memoria de 64 bits (FISTP)
11011111 oo110mmm disp	Memoria de 80 bits (FBSTP)

Formato	Ejemplos	Ciclos de reloj	
FST FSTP FIST FISTP FBSTP	FST DATO FST ST(3) FST FSTP FIST DATO2 FBSTP DATO6 FISTP DATO9	8087	15—540
		80287	15—540
		80387	11—534
		80486/7	3—176

FSTCW/FNSTCW Almacenar registro de control

11011001 oo111mmm disp

Formato	Ejemplos	Ciclos de reloj	
FSTCW FNSTCW	FSTCW CONTROL FNSTCW ESTADO FSTCW MAQUINA	8087	12—18
		80287	12—18
		80387	15
		80486/7	3

FSTENV/FNSTENV Almacenar ambiente

11011001 oo110mmm disp

Formato	Ejemplos	Ciclos de reloj	
FSTENV FNSTENV	FSTENV CONTROL FNSTENV ESTADO FSTENV MAQUINA	8087	40—50
		80287	40—50
		80387	103—104
		80486/7	58—67

FSTSW/FNSTSW Almacenar registro de estado

11011101 oo111mmm disp

Formato	Ejemplos	Ciclos de reloj	
FSTSW FNSTSW	FSTSW CONTROL FNSTSW ESTADO FSTSW MAQUINA	8087	12—18
		80287	12—18
		80387	15
		80486/7	3

FSUB/FSUBP/FISUB Restar

11011000 oo100mmm disp	Memoria de 32 bits (FSUB)
11011100 oo100mmm disp	Memoria de 64 bits (FSUB)
11011d00 11101rrr	FSUB ST,ST(rrr)
11011110 11101rrr	FSUBP ST,ST(rrr)
11011110 oo100mmm disp	Memoria de 16 bits (FISUB)
11011010 oo100mmm disp	Memoria de 32 bits (FISUB)

TABLA 12–6 (continuación)

Formato	Ejemplos	Ciclos de reloj	
FSUB FSUBP FISUB	FSUB DATO FSUB ST,ST(2) FSUB ST(2),ST FSUBP FISUB DATO3	8087	70—143
		80287	70—143
		80387	29—82
		80486/7	8—35

FSUBR/FSUBRP/FISUBR Resta inversa

11011000 oo101mmm disp	Memoria de 32 bits (FSUBR)
11011100 oo101mmm disp	Memoria de 64 bits (FSUBR)
11011d00 11100rrr	FSUBR ST,ST(rrr)
11011110 11100rrr	FSUBRP ST,ST(rrr)
11011110 oo101mmm disp	Memoria de 16 bits (FISUBR)
11011010 oo101mmm disp	Memoria de 32 bits (FISUBR)

Formato	Ejemplos	Ciclos de reloj	
FSUBR FSUBRP FISUBR	FSUBR DATO FSUBR ST,ST(2) FSUBR ST(2),ST FSUBRP FISUBR DATO3	8087	70—143
		80287	70—143
		80387	29—82
		80486/7	8—35

FTST Comparar ST(0) con + 0.0

11011001 11100100

Ejemplo	Ciclos de reloj	
FTST	8087	38—48
	80287	38—48
	80387	28
	80486/7	4

FUCOM/FUCOMP/FUCOMPP Comparación desordenada

11011101 11100rrr	FUCOM ST,ST(rrr)
11011101 11101rrr	FUCOMP ST,ST(rrr)
11011101 11101001	FUCOMPP

Formato	Ejemplos	Ciclos de reloj	
FUCOM FUCOMP FUCOMPP	FUCOM ST,ST(2) FUCOM FUCOMP ST,ST(3) FUCOMP FUCOMPP	8087	—
		80287	—
		80387	24—26
		80486/7	4—5

11011001 11110001

Ejemplo		Ciclos de reloj
FYL2X	8087	900—1100
	80287	900—1100
	80387	120—538
	80486/7	196—329

FXL2XP1 ST(1) x $\log_2$ [ST(0) + 1.0]

11011001 11111001

Ejemplo		Ciclos de reloj
FXL2XP1	8087	700—1000
	80287	700—1000
	80387	257—547
	80486/7	171—326

Nota: d = dirección, donde d = 0 para ST como el destino y d = 1 para ST como fuente, rrr = número de registro de punto decimal flotante, oo = modo, mmm = campo r/m, y disp = desplazamiento.

indican los tiempos de ejecución para el 8087, 80287, 80387 y 80486 o 80487. Para determinar el tiempo de ejecución de una instrucción se multiplica el ciclo del reloj por el número de ciclos de reloj que se requiere la de ejecución. La instrucción de FADD requiere de 70-143 ciclos de reloj para el 80287. Supongamos que se utiliza un reloj de 8 MHz para 80287 que tiene un periodo de reloj de 1/8 MHz o 125 ns por reloj. La instrucción de FADD requiere entre 8.75 us y 17.875 gms para ejecutarse. Si se utiliza un 80486 de 25 Mhz (40 ns), esta instrucción requiere entre 0.32 µs y 0.8 µs para ejecutarse.

La tabla 12-6 utiliza algunas anotaciones abreviadas para representar el desplazamiento que puede o no ser necesario para una instrucción que utiliza un modo de direccionamiento de memoria. También utiliza la abreviación para representar el modo, mmm, para representar un modo de direccionamiento de registro/memoria, y rrr para representar uno de los registros del coprocesador de punto flotante ST(0)-ST(7). El bit d que aparece en algunos de los códigos de instrucción define la dirección del flujo de datos como FADD ST,ST(2) o FADD ST(2),ST. El bit d es un 0 lógico para un flujo hacia ST como en FADD ST,ST(2) y un 1 lógico de 1 para FADD ST(2),ST.

12-5 PROGRAMACION DEL COPROCESADOR ARITMETICO

Esta sección del capítulo proporciona ejemplos de programación para el coprocesador aritmético. Cada ejemplo se escogió para ilustrar una técnica de programación para el coprocesador.

Cómo calcular el área de un círculo

Este ejemplo de programación proporciona una ilustración simple de un método de direccionamiento de la pila del coprocesador. Primero hay que recordar que la ecuación para calcular el área de un círculo es $A = \pi R^2$. En el ejemplo 12-6 se indica un procedimiento que realiza este cálculo.

EJEMPLO 12-6

```
                    ;Procedimiento que calcula el área de un círculo.
                    ;
                    ;El radio se debe guardar en la localidad RADIO de la memoria
                    ;antes de llamar a este procedimiento. El resultado se encuentra
                    ;en la localidad AREA de la memoria después del procedimiento.
                    ;
0000                AREAS   PROC    FAR

0000 D9 06 0004 R           FLD     RADIO           ;radio a ST
0004 D8 C8                  FMUL    ST,ST(0)        ;radio al cuadrado
0006 D9 EB                  FLDPI                   ;π a ST
0008 DE C9                  FMUL                    ;multiplicar ST = ST × ST(1)
000A D9 1E 0000 R           FSTP    AREA            ;guardar área
000E 9B                     FWAIT                   ;esperar coprocesador
000F CB                     RET

0010                AREAS   ENDP
```

Este es un procedimiento bastante sencillo, pero sí demuestra la operación de la pila. Para proporcionar un mejor entendimiento de la operación de la pila, la figura 12-12 muestra el contenido de la pila después de que se ejecuta cada instrucción del ejemplo 12-6.

La primera instrucción carga el contenido de la localidad de la memoria RADIUS a la parte alta de la pila. Enseguida la instrucción FMUL ST,ST(0) eleva al cuadrado el radio en la parte alta de la pila. La instrucción de FLDPI carga π a la parte alta de la pila. Las instrucciones de FMUL utilizan el modo clásico de direccionamiento de pila para multiplicar ST por ST(1). Después de la multiplicación, ambos valores anteriores se retiran de la pila y el producto los reemplaza en la parte alta de la pila. Finalmente la instrucción de FSTP copia la parte alta de la pila, el área, a la localidad de la memoria AREA y limpia la pila.

Cómo encontrar la frecuencia resonante

La fórmula para determinar la frecuencia de resonancia de un circuito LC es una ecuación que se encuentra comúnmente en la electrónica. La ecuación resuelta por el procedimiento ilustrado en el ejemplo 12-7 es $F = 1/(2\pi\sqrt{LC})$.

FWAIT Esperar

10011011

Ejemplo		Ciclos de reloj
FWAIT	8087	4
	80287	3
	80387	6
	80486/7	1—3

FXAM Examinar ST(0)

11011001 11100101

Ejemplo		Ciclos de reloj
FXAM	8087	12—23
	80287	12—23
	80387	30—38
	80486/7	8

FXCH Intercambiar ST(0) con otro registro

11011001 11001rrr FXCH ST,ST(rrr)

Formato	Ejemplos		Ciclos de reloj
FXCH	FXCH ST,ST(1) FXCH FXCH ST,ST(4)	8087	10—15
		80287	10—15
		80387	18
		80486/7	4

FXTRACT Extraer componentes de ST(0)

11011001 11110100

Ejemplo		Ciclos de reloj
FXTRACT	8087	27—55
	80287	27—55
	80387	70—76
	80486/7	16—20

FYL2X ST(1) x $\log_2$ ST(0)

FIGURA 12–12 Operación de la pila del 80287 con el procedimiento del ejemplo 12-6.
Nota: Se muestra la pila después de la ejecución de la instrucción indicada.

EJEMPLO 12–7

```
0000                    DATOS   SEGMENT

0000 00000000           RESO    DD      ?               ;frecuencia resonante
0004 358637BD           L       DD      .000001         ;inducción
0008 358637BD           C       DD      .000001         ;capacidad
000C 40000000           TWO     DD      2.0             ;constante

0010                    DATAS   ENDS

0000                    CODE    SEGMENT

                                ASSUME CS:Code,DS:DATAS

                        ;Procedimiento que encuentra la frecuencia resonante.

0000                    FREQ    PROC    FAR

0000 D9 06 0004 R               FLD     L               ;obtener     L
0004 D8 0E 0008 R               FMUL    C               ;encontrar   LC

0008 D9 FA                      FSQRT                   ;encontrar   √LC

000A D8 0E 000C R               FMUL    TWO             ;encontrar   2√LC
```

```
000E D9 EB                    FLDPI                ;obtener   π
0010 DE C9                    FMUL                 ;obtener   2π√LC

0012 D9 E8                    FLD1                 ;obtener   1
0014 DE F1                    FDIVR                ;forma     1/(2π√LC)

0016 D9 1E 0000 R             FSTP     RESO        ;guardar frecuencia

001A 9B                       FWAIT
001B CB                       RET

001C               FREQ       ENDP

001C               CODE       ENDS

                              END
```

Observe la forma directa y sencilla en que el procedimiento resuelve esta ecuación. Se requiere de muy poca manipulación adicional de los datos debido a que la pila está dentro del coprocesador. Observe también cómo el constante DOS se define para el programa y cómo se utiliza DIVRP para formar el recíproco por medio del uso del direccionamiento clásico de pila.

Cómo encontrar las raíces utilizando la ecuación cuadrática

Este ejemplo ilustra cómo encontrar las raíces de una expresión polinomial ($ax^2 + bx + c = 0$) utilizando la ecuación cuadrática. La ecuación cuadrática es ($b \pm \sqrt{\sqrt{b^2 - 4ac}}$)/2a. El ejemplo muestra un procedimiento que encuentra las raíces (R1 y R2) para la ecuación. Las constantes se almacenan en las localidades A, B, y C de la memoria.

EJEMPLO 12-8

```
                         .286
                         .287
0000                     DATAS    SEGMENT
0000 40000000            TWO      DD      2.0
0004 40800000            FOUR     DD      4.0
0008 3F800000            A        DD      1.0
000C C1800000            B        DD      -16.0
0010 421C0000            C        DD      +39.0
0014 00000000            R1       DD      ?
0018 00000000            R2       DD      ?
001C                     DATAS    ENDS

0000                     CODE     SEGMENT

                                  ASSUME CS:CODE,DS:DATAS

                         ;Procedimiento que resuelve la ecuación cuadrática.

0000                     ROOTS    PROC    FAR

0000 D9 06 0000 R                 FLD     TWO
0004 D8 0E 0008 R                 FMUL    A               ;forma 2a
0008 D9 06 0004 R                 FLD     FOUR
000C D8 0E 0008 R                 FMUL    A
0010 D8 0E 0010 R                 FMUL    C               ;forma 4ac
```

```
0014 D9 06 000C R          FLD    B
0018 D8 0E 000C R          FMUL   B              ;forma b²
001C DE E1                 FSUBR                 ;forma b² - 4ac
001E D9 FA                 FSQRT                 ;forma de raiz cuadrada de b² - 4ac

0020 D9 06 000C R          FLD    B
0024 D8 E1                 FSUB   ST,ST(1)
0026 D8 F2                 FDIV   ST,ST(2)
0028 D9 1E 0014 R          FSTP   R1             ;guardar raiz1

002C D9 06 000C R          FLD    B
0030 DE C1                 FADD
0032 DE F1                 FDIVR
0034 D9 1E 0018 R          FSTP   R2             ;guardar raiz2

0038 9B                    FWAIT
0039 CB                    RET

003A              ROOTS    ENDP

003A              CODE     ENDS
                           END
```

Cómo exhibir un número con punto decimal flotante de precisión sencilla

Esta sección del texto muestra cómo tomar el contenido del punto decimal flotante de un número con punto decimal flotante de precisión sencilla de 32 bits y mostrarlo en la pantalla de video. El procedimiento muestra el número con punto decimal flotante como un número mixto con una parte de entera y una parte fraccional separado por un punto decimal. Para poder simplificar el procedimiento, hemos colocado un límite al tamaño de la presentación del número mixto para que la porción del número entero sea un número binario de 32 bits y la fracción sea un número binario de 24 bits. El procedimiento no funcionará adecuadamente para números más grandes o más pequeños.

El ejemplo 12-9 indica el procedimiento para mostrar el contenido de la localidad NUMB de la memoria en la pantalla de video en la posición actual del cursor. El procedimiento primero prueba el signo del número y muestra un espacio para positivo y un signo de menos para un número negativo. Después de mostrar el signo, la instrucción de FABS convierte en positivo el número.

EJEMPLO 12–9

```
                  .286
                  .287
0000              DATAS    SEGMENT

0000 C50B0C00     NUMB     DD     -2224.75
0004 0000         TEMP     DW     ?
0006 0000         WHOLE    DW     ?
0008 00000000     FRACT    DD     ?

000C              DATAS    ENDS

0000              CODE     SEGMENT
```

```
                        ASSUME  CS:CODE,DS:DATAS

                 ;Programa principal que muestra NUMB
                 ;
0000             MAIN   PROC    FAR

0000 B8 ---- R          MOV     AX,DATAS
0003 8E D8              MOV     DS,AX
0005 E8 0013 R          CALL    DISP              ;muestra NUMB
0008 B4 4C              MOV     AH,4CH            ;salir del DOS
000A CD 21              INT     21H

000C             MAIN   ENDP

000C             DISPS  PROC    NEAR

000C B4 06              MOV     AH,6              ;mostrar AL
000E 8A D0              MOV     DL,AL
0010 CD 21              INT     21H
0012 C3                 RET

0013             DISPS  ENDP

0013             DISP   PROC    NEAR

0013 9B D9 3E 0004 R    FSTCW   TEMP              ;inicializar redondeo para recortar
0018 81 0E 0004 R 0C00  OR      TEMP,0C00H
001E 9B D9 2E 0004 R    FLDCW   TEMP

0023 D9 06 0000 R       FLD     NUMB              ;obtener NUMB
0027 D9 E4              FTST                      ;probar NUMB
0029 9B DF E0           FSTSW   AX                ;estado a AX
002C 25 4500            AND     AX,4500H          ;obtener C3, C2 y C0
002F 3D 0100            CMP     AX,0100H          ;probar para -
0032 75 05              JNE     DISP1             ;si es positivo
0034 B0 2D              MOV     AL,'-'
0036 E8 000C R          CALL    DISPS             ;mostrar menos

0039             DISP1:

0039 D9 E1              FABS                      ;convertir ST en positivo
003B D9 FC              FRNDINT                   ;obtener número entero
003D DF 16 0006 R       FIST    WHOLE             ;guardar número entero
0041 D9 06 0000 R       FLD     NUMB
0045 D9 E1              FABS
0047 DE E9              FSUB                      ;obtener fracción
0049 D9 E1              FABS
004B D9 1E 0008 R       FSTP    FRACT             ;guardar fracción
004F 9B                 FWAIT

                 ;mostrar parte de número entero

0050 A1 0006 R          MOV     AX,WHOLE
0053 B9 0000            MOV     CX,0
0056 BB 000A            MOV     BX,10

0059             DISP2:

0059 41                 INC     CX
005A 33 D2              XOR     DX,DX
005C F7 F3              DIV     BX
```

```
005E 83 C2 30              ADD    DX,'0'              ;convertir a ASCII
0061 52                    PUSH   DX
0062 0B C0                 OR     AX,AX
0064 75 F3                 JNE    DISP2               ;si no es cero

0066              DISP3:

0066 58                    POP    AX
0067 E8 000C R             CALL   DISPS               ;mostrarlo
006A E2 FA                 LOOP   DISP3
006C B0 2E                 MOV    AL,'.'              ;mostrar punto decimal
006E E8 000C R             CALL   DISPS

                  ;mostrar parte de fracción

0071 A1 0008 R             MOV    AX,WORD PTR FRACT
0074 8B 16 000A R          MOV    DX,WORD PTR FRACT+2
0078 B9 0008               MOV    CX,8

007B              DISP4:

007B D1 E0                 SHL    AX,1
007D D1 D2                 RCL    DX,1
007F E2 FA                 LOOP   DISP4
0081 81 CA 8000            OR     DX,8000H            ;instalar bit implícito
0085 92                    XCHG   AX,DX
0086 BB 000A               MOV    BX,10

0089              DISP5:

0089 F7 E3                 MUL    BX
008B 50                    PUSH   AX
008C 92                    XCHG   DX,AX
008D 04 30                 ADD    AL,'0'
008F E8 000C R             CALL   DISPS               ;mostrar dígito
0092 58                    POP    AX
0093 0B C0                 OR     AX,AX
0095 75 F2                 JNZ    DISP5
0097 C3                    RET

0098              DISP     ENDP

0098              CODE     ENDS

                           END    MAIN
```

La última parte del procedimiento muestra la parte entera del número seguido por la parte fraccional. Observe que la parte fraccional puede contener un error de redondeo para ciertos valores. No hemos ajustado el número para quitar el error de redondeo que es inherente en los números fraccionales de punto decimal flotante.

Cómo leer del teclado un número mixto

Si se utiliza aritmética de punto flotante en un programa, debemos contar con un método para leer el número del teclado y convertirlo a punto flotante. El procedimiento indicado en el ejemplo

12-10 lee del teclado un número mixto con signo y lo convierte en un número de punto flotante localizado en la parte alta de la pila dentro del coprocesador.

EJEMPLO 12–10

```
                        .286
                        .287
0000                    DATA    SEGMENT

0000 00                 SIGN    DB      ?
0001 0000               TEMP1   DW      ?
0003 41200000           TEN     DD      10.0

0007                    DATA    ENDS

0000                    CODE    SEGMENT

                                ASSUME CS:CODE,DS:DATA

                        ;procedimiento que lee un número mixto del teclado
                        ;y lo deja en la parte alta de la pila del coprocesador.

0000                    READ    PROC    FAR

0000 B8 ---- R                  MOV     AX,DATA         ;segmento de datos de direccionamiento
0003 8E D8                      MOV     DS,AX
0005 9B D9 EE                   FLDZ                    ;borrar ST
0008 C6 06 0000 R 00            MOV     SIGN,0          ;borrar signo
000D E8 007C                    CALL    GET             ;leer un carácter
0010 3C 2D                      CMP     AL,'-'          ;probar para menos
0012 75 07                      JNE     READ1           ;si no es menos
0014 C6 06 0000 R FF            MOV     SIGN,0FFH       ;instalar signo para menos
0019 EB 0F                      JMP     READ3           ;obtener parte del número entero

001B                    READ1:

001B 3C 2B                      CMP     AL,'+'          ;probar para más
001D 74 0B                      JE      READ3           ;obtener parte del número entero
001F 3C 30                      CMP     AL,'0'          ;probar para número
0021 72 06                      JB      READ2
0023 3C 39                      CMP     AL,'9'
0025 77 02                      JA      READ2
0027 EB 04                      JMP     READ4           ;si es número

0029                    READ2:

0029 CB                         RET

002A                    READ3:

002A E8 005F                    CALL    GET             ;leer parte del número entero

002D                    READ4:

002D 3C 2E                      CMP     AL,'.'          ;probar para fracción
002F 74 27                      JE      READ7           ;si es fracción
0031 3C 30                      CMP     AL,'0'          ;probar para número
0033 72 17                      JB      READ5
0035 3C 39                      CMP     AL,'9'
```

```
0037 77 13                    JA      READ5
0039 9B D8 0E 0003 R          FMUL    TEN          ;formar número entero
003E 32 E4                    XOR     AH,AH
0040 2C 30                    SUB     AL,'0'
0042 A3 0001 R                MOV     TEMP1,AX
0045 9B DE 06 0001 R          FIADD   TEMP1
004A EB DE                    JMP     READ3

004C                  READ5:

004C 80 3E 0000 R 00          CMP     SIGN,0       ;ajustar signo
0051 75 01                    JNE     READ6
0053 CB                       RET

0054                  READ6:

0054 9B D9 E0                 FCHS
0057 CB                       RET

0058                  READ7:

0058 9B D9 E8                 FLD1                 ;de fracción
005B 9B D8 36 0003 R          FDIV    TEN

0060                  READ8:

0060 E8 0029                  CALL    GET          ;leer carácter
0063 3C 30                    CMP     AL,'0'       ;probar para número
0065 72 20                    JB      READ9
0067 3C 39                    CMP     AL,'9'
0069 77 1C                    JA      READ9
006B 32 E4                    XOR     AH,AH
006D 2C 30                    SUB     AL,'0'
006F A3 0001 R                MOV     TEMP1,AX
0072 9B DF 06 0001 R          FILD    TEMP1        ;cargar número
0077 9B D8 C9                 FMUL    ST,ST(1)     ;forma de fracción
007A 9B DC C2                 FADD    ST(2),ST
007D 9B D8 D9                 FCOMP
0080 9B D8 36 0003 R          FDIV    TEN
0085 EB D9                    JMP     READ8

0087                  READ9:

0087 9B D8 D9                 FCOMP                ;limpiar pila
008A EB C0                    JMP     READ5

008C                  READ    ENDP

008C                  GET     PROC    NEAR

008C B4 06                    MOV     AH,6         ;leer carácter
008E B2 FF                    MOV     DL,0FFH
0090 CD 21                    INT     21H
0092 74 F8                    JZ      GET
0094 C3                       RET

0095                  GET     ENDP

0095                  CODE    ENDS

                              END
```

Aquí se lee primero el signo del teclado, si es que está presente, y luego se guarda para su uso posterior en el ajuste del signo del número de punto flotante que resulta. En seguida, se lee la parte entera del número. Esta parte termina con un punto, un espacio, o un retorno de carro. Si se escribe un punto, entonces el procedimiento continúa y lee la parte fraccional. Si se escribe un espacio o un retorno de carro, el número se convierte a la forma de punto flotante.

12–6 RESUMEN

1. Las funciones del coprocesador aritmético funcionan de manera paralela con el microprocesador.
2. Los tipos de datos manipulados por el coprocesador incluyen números enteros con signo, punto decimal flotante, y decimal codificado en binario (BCD).
3. Hay tres formas de números enteros para el coprocesador: palabra (16 bits), corto (32 bits), y largo (64 bits). Cada número entero contiene un número con signo en su magnitud real para los números positivos y en forma de complemento a dos para los números negativos.
4. Un número BCD se almacena como un número de 18 dígitos en 10 bytes de memoria. El byte más significativo contiene el bit de signo, y los 9 bytes restantes contienen un número BCD empacado en 18 dígitos.
5. El coprocesador soporta tres tipos de números con punto decimal flotante: precisión sencilla (32 bits), doble precisión (64 bits), y precisión extendida (80 bits). Un número con punto decimal flotante está formado de tres partes: el signo, el exponente polarizado, y la mantisa. En el coprocesador, el exponente está polarizado con una constante y el bit de número entero del número normalizado no está guardado en la mantisa excepto en la forma de posición extendida.
6. Los números decimales se convierten a punto flotante convirtiendo el número a binario, normalizando el número binario, agregando la polarización al exponente, y guardando el número en forma de punto decimal flotante.
7. Los números de punto flotante se convierten a decimal restando la polarización del exponente, desnormalizando el número, y luego convirtiéndolo a decimal.
8. El 80287 utiliza espacio de entrada/salida para la ejecución de algunas de sus instrucciones. Este espacio es invisible para el programa y se utiliza internamente en el sistema 80286/80287. Estas direcciones de entrada/salida de 16 bits son 00F8H, 00FAH y 00FCH, que no se deben utilizar para transferencias de datos de entrada/salida en un sistema que contiene un 80287.
9. El coprocesador contiene un registro de estado que indica ocupado, qué condiciones siguen de una comparación o prueba, la localización de la parte alta de la pila, y el estado de los bits de error.
10. El registro de control del coprocesador contiene bits de control que seleccionan infinito, redondeo, precisión y enmascaramiento de errores.

11. Los siguientes directivos frecuentemente se utilizan con el coprocesador para guardar datos: DW (definir palabra), DD (definir doble palabra), DQ (definir cuádruple palabra) y DT (definir 10 bytes).
12. El coprocesador utiliza una pila para transferir datos entre él mismo y el sistema de memoria. Por lo general los datos se cargan o se recuperan de la parte alta de la pila para su almacenamiento.
13. Todos los datos internos del coprocesador siempre están en la forma de precisión extendida de 80 bits. El único momento en que los datos se encuentran en cualquier otra forma es cuando se almacenan o se cargan de la memoria.
14. Los modos para el direccionamiento del coprocesador incluyen: el modo clásico de pila, registro, registro con recuperación, y memoria. El direccionamiento de pila es implícito y los datos en ST se convierten en la fuente, ST(1) el destino, y el resultado se encuentra en ST después de una recuperación. Los otros modos para el direccionamiento se explican por sí mismos.
15. Las operaciones aritméticas del coprocesador incluyen suma, resta, multiplicación, división y raíz cuadrada.
16. En el conjunto de instrucciones del coprocesador hay funciones trascendentales. Estas funciones encuentran la tangente parcial o arco tangente, $2^X - 1$, Y $\log_2 X$ y Y $\log_2(X + 1)$.
17. Se almacenan dentro del coprocesador constantes que proporcionan: +0.0, +1.0, π, $\log_2 10$, $\log_2 e$, $\log_{10} 2$, y $\log_e 2$.
18. El 80387 funciona con el microprocesador 80386 y el 80487SX funciona con el microprocesador 80486SX, pero el 80486DX contiene su propio coprocesador aritmético interno. Las instrucciones realizadas por las versiones anteriores están disponibles en estos coprocesadores. Además de estas instrucciones, el 80387 y el 80486/7 también pueden calcular el seno y coseno.

12–7 PREGUNTAS Y PROBLEMAS

1. Indique los tres tipos de datos que se cargan o se almacenan de la memoria al coprocesador.
2. Indique los tres tipos de datos para números enteros, el campo de los números enteros guardados en ellos, y el número de bits designado para cada uno.
3. Explique cómo se almacena en la memoria un número en BCD el coprocesador.
4. Indique los tres tipos de números con punto decimal flotante utilizados con el coprocesador y el número de bits binarios asignados para cada uno.
5. Convierta los siguientes números decimales a números con punto decimal flotante de precisión sencilla:

   a. 28.75 b. 624 c. −0.615 d. +0.0 e. −1000.5
6. Convierta los siguientes números con punto decimal flotante de precisión sencilla a decimales:

   a. 11000000 11110000 00000000 00000000

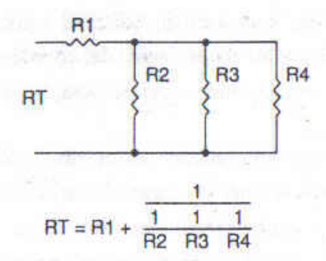
   b. 00111111 00010000 00000000 00000000

c. 01000011 10011001 00000000 00000000
d. 01000000 00000000 00000000 00000000
e. 01000001 00100000 00000000 00000000
f. 00000000 00000000 00000000 00000000

7. ¿Cuál es el propósito de la terminal CLM en el 80287?
8. ¿Dónde está conectada la terminal $\overline{BUSY}$ en el 80287?
9. ¿Cuál es el propósito de $\overline{NPRD}$ y dónde se conecta?
10. Explique lo que hace el 80287 cuando se ejecuta una instrucción normal del 80286.
11. Explique qué hace el 80286 cuando se ejecuta una instrucción del 80287.
12. ¿Cuál es el propósito de los bits C3–C0 en el registro de estado?
13. ¿Cómo se selecciona el modo de redondeo en el 80287?
14. ¿Qué instrucción del coprocesador utiliza el registro AX del microprocesador?
15. ¿Cómo se almacenan los datos dentro del coprocesador?
16. Cada vez que el coprocesador se reinicializa, la parte alta del registro de la pila es el registro número __________.
17. ¿Que significa el término recortar en los bits de control de redondeo del registro de control?
18. ¿Cuál es la diferencia entre el control del infinito de afinidad y proyectivo?
19. ¿Qué instrucción del microprocesador forma el código de funcionamiento para el coprocesador?
20. Utilizando seudo códigos de instrución del ensamblador, forme instrucciones que realicen lo siguiente:
    a. Almacene un 23.44 en una localidad de memoria de punto decimal flotante de precisión doble llamada RANA.
    b. Guarde –123 en una localidad de número entero con signo de 32 bits llamada DATO3.
    c. Guarde –23.8 en una localidad de memoria de punto decimal flotante de precisión sencilla llamada DATO1.
    d. Reserve una localidad de memoria de doble precisión llamada DATO2.
21. Describa cómo funciona la instrucción FST DATO. Suponga que DATO está definida como una localidad de memoria de 64 bits.
22. ¿Qué realiza la instrucción FILD DATO?
23. Forme una instrucción que sume el contenido del registro 3 a la parte alta de la pila.
24. Describa la operación de la instrucción FADD.
25. Seleccione una instrucción que reste el contenido del registro 2 de la parte alta de la pila y guarde el resultado en el registro 2.
26. ¿Cuál es la función de la instrucción FBSTP DATO?
27. ¿Cuál es la diferencia entre una división hacia delante y una inversa?
28. ¿Cuál es la diferencia entre la instrucción FTST y FXAM?
29. Explique lo que calcula la instrucción F2XM1.
30. ¿Qué instrucción transfiere π a la parte alta de la pila?
31. ¿Qué realizará FFREE ST(2) cuando se ejecute?
32. ¿Qué instrucción guarda el ambiente?

33. ¿Qué guarda la instrucción FSAVE?
34. Desarrolle un procedimiento que encuentre el área de un rectángulo (A = L × W). Las localidades de la memoria para este procedimiento son A, L y W de precisión sencilla.
35. Escriba un procedimiento que encuentre la reactancia inductiva (XL = 2 π FL). Las localidades de la memoria para este procedimiento son XL, F y L de precisión sencilla.
36. Desarrolle un procedimiento que genere una tabla de raíz cuadrada para los números enteros del 0 al 10.
37. ¿Cuándo se utiliza la instrucción FWAIT?
38. Con el circuito serie/paralelo y la ecuación ilustrada en la figura 12-13, desarrolle un programa utilizando valores de precisión sencilla para R1, R2, R3 y R4, que encuentra la resistencia total y guarda el resultado en una localidad RT de precisión sencilla.

FIGURA 12–13



$$RT = R1 + \frac{1}{\frac{1}{R2} + \frac{1}{R3} + \frac{1}{R4}}$$

CAPITULO 13

El microprocesador 80186/80188 y 80286

INTRODUCCION

Los Intel 80186/80188 y 80286 son las versiones mejoradas del microprocesador anterior 8086/8088. Los 80186/80188 y 80286 son todos microprocesadores de 16-bits que son compatibles al 8086/8088. Aun la arquitectura de estos microprocesadores es semejante a la de las versiones anteriores. Este capítulo presenta un repaso general de cada microprocesador y señala las diferencias o mejoras que aparecen en cada versión. La primera parte del capítulo describe el microprocesador 80186/80188, y la última parte muestra el microprocesador 80286.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Describir las mejorías en arquitectura y programación de los microprocesadores 80186/80188 y 80286 comparados con el 8086/8088.
2. Establecer la interface de los 80186/80188 y 80286 a la memoria y E/S.
3. Desarrollar programación utilizando las mejoras proporcionadas por estos microprocesadores.
4. Describir la operación de la unidad de administración de memoria (MMU) dentro del microprocesador 80286.

13–1 ARQUITECTURA DEL 80186/80188

Los 80186 y 80188, así como los 8086 y 8088, son casi idénticos. La única diferencia entre el 80186 y 80188 es el ancho de sus canales para datos. El 80186 (como el 8086) contiene un canal de datos de 16 bits, mientras que el 80188 (como el 8088) tiene un canal para datos de 8 bits. La estructura de los registros internos del 80186/80188 es casi idéntica a la del 8086/8088. Casi la

única diferencia es que el 80186/80188 contiene vectores de interrupción adicionales reservados y algunas muy poderosas características de E/S integradas E/S. Los 80186/80188 son llamados a menudo controladores dedicados debido a su aplicación, que no es como una computadora basada en un microprocesador, sino como un controlador.

Diagrama a bloques del 80186

La figura 13-1 proporciona el diagrama a bloques del microprocesador 80186. Observe que este microprocesador tiene más circuitos internos que el 8086. El diagrama a bloques del 80186 y 80188 es idéntico, excepto por la cola de espera, la cual es de 4 bytes en el 80188 y de 6 bytes en el 80186. Como el 8086, el 80186 contiene una unidad de interface del canal (BIU) y una unidad de ejecución (EU).

Además de la BIU y EU, el 80186 contiene un generador de reloj, un controlador programable de interrupciones, temporizadores programables, un controlador de DMA programable y una unidad de señales de habilitación programable. Estas mejorías aumentan mucho la utilidad del 80186 y reducen el número de componentes periféricos requeridos para implantar un sistema. Muchos subsistemas populares para la computadora personal utilizan el microprocesador 80186

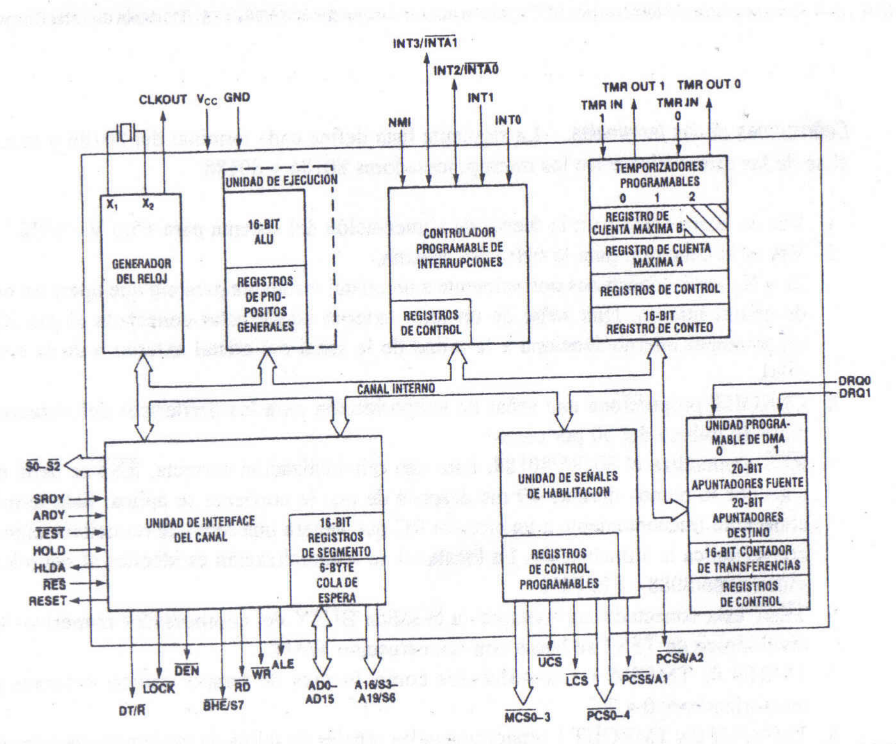


FIGURA 13-1 El diagrama a bloques del microprocesador 80186. Observe que todo el diagrama en bloque del 80188 es idéntico excepto que faltan $\overline{BHE}$/S7 y AD15-AD8 que están etiquetados como A15-A8. (Cortesía de Intel Corporation.)

como controladores de discos, controladores para una red de área local (LAN), etc. El 80186 también se aplica en la red de teléfonos celulares como conmutador.

La programación para el 80186/80188 es idéntica a la del microprocesador 80286 sin las instrucciones para la administración de la memoria. Esto significa que la multiplicación, corrimientos con cuenta inmediata, cadenas de E/S, PUSHA, POPA, BOUND y ENTER, así como LEAVE, funcionan en el microprocesador 80186/80188.

Las mejoras en los 80186/80188

En este segmento del texto, introduciremos los cambios del microprocesador 80186/80188 o controlador dedicado, pero no proporcionaremos una explicación exclusiva. Más detalles sobre el funcionamiento de cada cambio se proporcionarán más adelante en este capítulo.

Generador del reloj. El generador del reloj interno reemplaza al generador externo de reloj 8284A externo utilizado con el microprocesador 8086/8088. Esto reduce el número de componentes en un sistema.

El generador del reloj interno tiene tres terminales de conexión: X_1, X_2, y CLKOUT. Las terminales X_1 y X_2 se conectan a un cristal que resonará al doble de la frecuencia de operación del microprocesador. En la versión de 8 MHz del 80186/80188, un cristal de 16 MHz está conectado a X_1 y X_2. El 80186/80188 está disponible en versiones de 6 MHz, 8 MHz, 12 MHz, o 16 MHz y también en CMOS como el 80C186 y 80C188.

La terminal CLKOUT proporciona una señal del reloj al sistema que es la mitad de la frecuencia del cristal con un ciclo de trabajo del 50 por ciento. La terminal CLKOUT se conecta a otros dispositivos en un sistema y proporciona una fuente de temporización a microprocesadores adicionales en el sistema.

Además de estas terminales externas, el generador del reloj proporciona la temporización interna para poder sincronizar la terminal de entrada READY, mientras que en un sistema 8086/8088, la sincronización de READY la proporciona el generador de reloj 8284A.

Controlador programable de interrupciones. El controlador programable de interrupciones (PCI) es el "árbitro" todas las interrupciones internas y externas, y controla hasta dos PCIs 8259A externos. Cuando se agrega un 8259, el 80186/80188 funciona como el amo y el 8259 opera como el esclavo.

Si el PCI se opera sin un 8259 externo, tiene cinco entradas para interrupciones: INT_0-INT_3 y NMI. Esta es una expansión de las dos entradas para interrupciones disponibles en el microprocesador 8086/8088. En muchos sistemas, las cinco entradas para los interruptores son suficientes.

Temporizadores. La sección de temporización contiene tres temporizadores de 16 bits totalmente programables. Los temporizadores 0 y 1 generan formas de ondas para uso externo y son manejadas ya sea por el reloj del 80186/80188 o por un reloj externo. También son utilizados para contar los procesos externos. El tercer temporizador, el 2, es interno y está conectado al reloj principal. La salida del 2 generará una interrupción después de un número específico de ciclos de reloj y también puede proporcionar pulsos de reloj a los otros temporizadores. El temporizador 2 también puede utilizarse como un reloj para vigilar (watchdog), debido a que se puede programar para interrumpir al microprocesador después de cierta cantidad de tiempo.

Unidad programable de DMA. La unidad programable de DMA contiene dos canales de DMA. Cada canal puede transferir datos entre localidades de la memoria, entre la memoria y las E/S, o entre los periféricos de E/S. Este controlador de DMA es semejante al controlador 8237 de DMA discutido en el Capítulo 12. La principal diferencia es que el 8237 tiene 4 canales.

Unidad de señales de habilitación programable. Esta unidad consta de una memoria programable y un decodificador de E/S programable. Tiene seis líneas de habilitación para memoria y siete líneas de habilitación para E/S.

Las líneas de habilitación para la memoria están divididas en tres grupos que seleccionan memoria para las secciones principales del mapa de la memoria del 80186/80188. La señal para selección de la memoria baja habilita a la memoria para los vectores de interruptores, la señal para la selección de la memoria superior habilita a la memoria para reinicialización, y las señales para seleccionar la memoria media habilitan hasta cuatro dispositivos de memoria. El límite de la memoria inferior comienza en la localidad 00000H, y el límite de la memoria superior termina en la localidad FFFFFH. El tamaño de las áreas de la memoria es programable, y los ciclos de espera (0-3 esperas) pueden insertarse automáticamente con la selección de un área de la memoria.

Cada señal de habilitación programable E/S direcciona a un bloque de 128-bytes del espacio de E/S. El área programable de E/S comienza en la dirección base de E/S programable por el usuario y todos los siete bloques de 128 bytes son contiguos.

Diagrama de base

La figura 13-2 ilustra el diagrama de base del microprocesador 80186. Observe que el 80186 está encapsulado ya sea en un encapsulado "sin terminales" (LCC) de 68 terminales como se muestra, o en rejilla de terminales (PGA). El encapsulado LCC se ilustra en la figura 13-3.

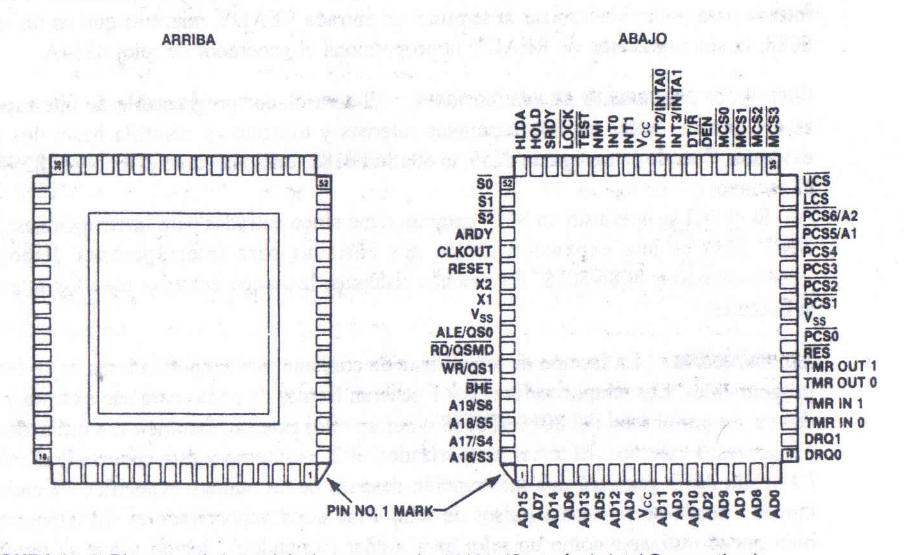


FIGURA 13-2 Diagrama de base del microprocesador 80186. (Cortesía de Intel Corporation.)

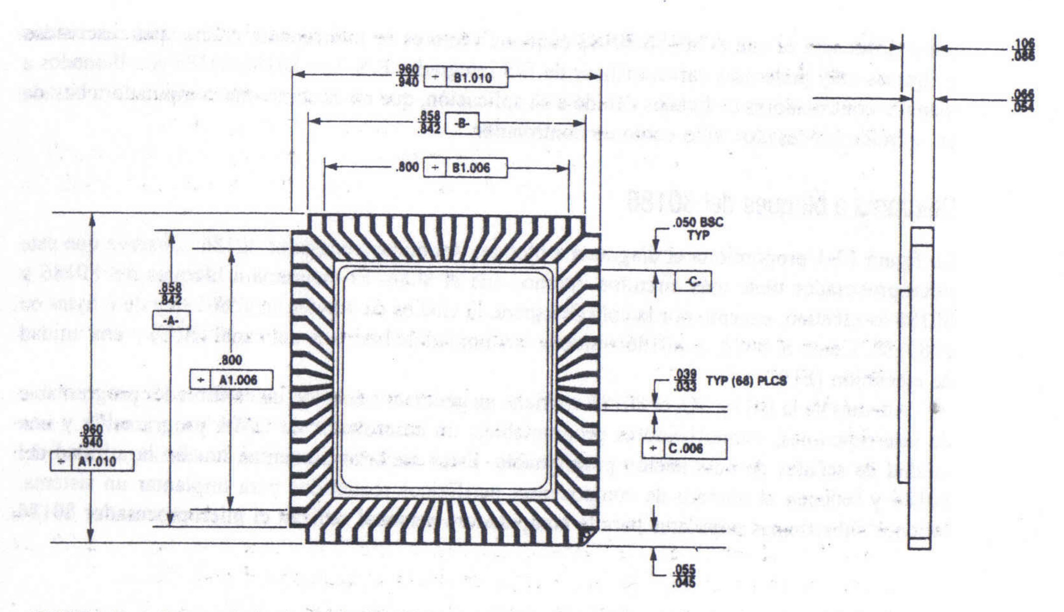


FIGURA 13–3 Encapsulado sin terminales (LCC) para el microprocesador 80186/80188. (Cortesía de Intel Corporation.)

Definiciones de las terminales. La siguiente lista define cada terminal del 80186 y muestra muchas de las diferencias entre los microprocesadores 80186 y 80188.

1. Vcc es la conexión para la fuente de alimentación del sistema para +5.0 V, +10%.
2. Vss es la conexión para la tierra del sistema.
3. X_1 y X_2 están conectados normalmente a un cristal resonante paralelo que opera un oscila-dor de cristal interno. Una señal de un reloj externo puede estar conectada al pin X_1. El reloj principal interno funciona a la mitad de la señal del cristal externo o de la entrada del reloj.
4. CLKOUT proporciona una señal de temporización para los periféricos del sistema, con un ciclo de trabajo del 50 por ciento.
5. $\overline{\text{RES}}$ reinicializa el 80186/80188. Para una reinicialización correcta, $\overline{\text{RES}}$ se debe mantener bajo por lo menos durante 50 ms después de que la corriente se aplica. Esta terminal está conectada frecuentemente a un circuito RC que genera una señal de reinicialización después que se aplica la alimentación. La localidad de reinicialización es idéntica al del microprocesador 8086/8088—FFFF0H.
6. $\overline{\text{TEST}}$ esta conectado normalmente a la salida BUSY del coprocesador numérico 80187. El nivel lógico de $\overline{\text{TEST}}$ se busca con la instrucción WAIT.
7. TMRIN 0, TMRIM 1 son utilizados como fuentes de temporización externas para los temporizadores 0 y 1.
8. TMROUT 0 y TMROUT 1 proporcionan las señales de salida de los temporizadores 0 y 1, los cuales pueden ser programados para que proporcionen ondas cuadradas o pulsos.

9. DRQ_0 y DRQ_1 son líneas de solicitud de ciclos de DMA activas en nivel alto para los canales 0 y 1 de DMA.
10. NMI es la entrada de las interrupciones no enmascarables. Es disparada por flanco positivo y siempre está activa. Cuando NMI está activa, utiliza el vector interrupciones 2.
11. INT_0, INT_1, $INT_2/INTA_0$, Y $INT_3/INTA_1$ son las entradas de las interrupciones enmascarables. Son activas en alto y es posible programarlas para que sean activas por flanco o nivel. Son configuradas como cuatro entradas de interrupción si ningún 8259 externo está presente o como 2 entradas de interrupciones si están presentes 8259.
12. A19/S6, A18/S5, A17/S4, A16/S3 son conexiones multiplexadas para que proporcionen las direcciones (A19-A16) y el estado (S6-S3). El bit del estado de S6 indica un ciclo del procesador cuando está en nivel bajo o un ciclo de DMA cuando está en alto. Los bits de estado restantes siempre son un 0 lógico.
13. AD15-AD0 son conexiones multiplexadas del canal de direcciones/datos. Durante T1, el 80186 coloca la dirección A15-A0 en estas terminales, y durante T2, T3 y T4, el 80186 las utiliza como canal de datos con las señales D15-D0. Observe que el 80188 tiene terminales AD7-AD0 y A15-A8.
14. $\overline{BHE}$/S7 indica (cuando tiene un 0 lógico) qué se están transfiriendo datos válidos por las terminales del canal de datos D15-D8. En el 80188, esta terminal está etiquetado S7. S7 siempre es un 1 lógico así que no se requiere ningún circuito para demultiplexar S7.
15. ALE/QS_0 es una salida miltiplexada para la salida de datos que contiene ALE medio ciclo de reloj antes que en el 8086. QS_0 es la señal del estado de la cola de espera requerida por el coprocesador 80187.
16. $\overline{WR}/QS_1$ causa que la información se escriba en la memoria o en E/S. La señal $\overline{WR}$ está activa durante T3, TW, y T4, y QS_1 está activo durante el modo de estado de la cola de espera y se proporciona para el 80187.
17. $\overline{RD}$/QSMD causa que se lea información de la memoria o E/S. La señal $\overline{RD}$ está activa durante T2, TW, y T4, y QSMD se selecciona si esta terminal de entrada/salida es llevada a tierra por el 80187. El único momento en que QS_0 y QS_1 están disponibles, es durante el modo de estado de la cola de espera.
18. ARDY (entrada asincrón de READY) informa al 80186/80188 que la memoria o E/S está lista para que el 80186/80188 lea o escriba la información. Si se conecta a +5.0 V, el microprocesador funciona normalmente; si está a tierra, el microprocesador inserta a los estados de espera.
19. SRDY (entrada sincrón de READY) está sincronizado con el reloj del sistema para proporcionar la temporización correcta para la terminal READY. Como ARDY, SRDY si se conecta a +5.0 V no hay estados de espera.
20. $\overline{LOCK}$ es una salida controlada por el prefijo LOCK. Si una instrucción tiene el prefijo $\overline{LOCK}$, esta terminal se convierte en un 0 lógico por la duración de la instrucción con el prefijo LOCK.
21. $\overline{S2}$, $\overline{S1}$, y $\overline{S0}$ son bits de estado que proporcionan el tipo actual de transferencia del canal del sistema. $\overline{S2}$ se utiliza como la señal $M/\overline{IO}$ y $\overline{S1}$ se usa como $DT/\overline{R}$. El 80188 utiliza la señal S2 como $M/\overline{IO}$ y no $IO/\overline{M}$ como en el 8088.
22. HOLD y HLDA son utilizadas con dispositivos externos de DMA. HOLD es la entrada de solicitud de un ciclo de DMA y HLDA es una señal de reconocimiento a la solicitud de DMA está activa. Durante los ciclos de DMA, el microprocesador flota sus canales de direcciones, datos, y control.

23. $\overline{UCS}$ (habilitación de la parte superior de la memoria) selecciona la parte superior del mapa de memoria. Esta salida es programable para permitir los tamaños de memoria de 1K-256K bytes terminando en la localidad FFFFFH.
24. $\overline{LCS}$ (habilitación de la parte inferior de la memoria) habilita a la memoria comenzando en la localidad 00000H. Esta terminal está programada para seleccionar los tamaños de la memoria desde 1K a 256K bytes.
25. $\overline{MCS_0}$-$\overline{MCS_3}$ (habilitación de la memoria media) habilita cuatro dispositivos en la parte media de la memoria. Estas terminales son programables para seleccionar un bloque de memoria de 8K-512K bytes contenido en cuatro dispositivos.
26. $\overline{PCS_0}$-$\overline{PCS_4}$ son cinco líneas diferentes de selección de periféricos.
27. $\overline{PCS_5}$/A1 y $\overline{PCS_6}$/A2 están programados como líneas de selección de periféricos o como bits de dirección con retención interna A2 y A1.
28. DT/$\overline{R}$ controla la dirección de los transceptores del canal de datos ligados del sistema en caso de que los haya.
29. $\overline{DEN}$ habilita a los registros externos del canal de datos.

Características de operación de CD

Es necesario saber las características de operación de CD antes de intentar conectar u operar el microprocesador. El microprocesador 80186/80188 requiere entre 450 y 550 mA de corriente y el 80C186/80C188 como 10 mA. Cada terminal de salida proporciona 2.0 mA o corriente lógica 0 y 400 μA de corriente lógica 1, excepto CLKOUT que proporciona 4.0 mA de corriente lógica 0 y los bits de estado que proporcionan 2.5 mA.

Temporización del 80186/80188

El diagrama de temporización del 80186 se proporciona en la figura 13-4. Para el 80188 es idéntico excepto por las conexiones para la dirección multiplexada, las cuales son AD7-AD0 en vez de AD15-AD0, y el BHE, el cual no existe en el 80188.

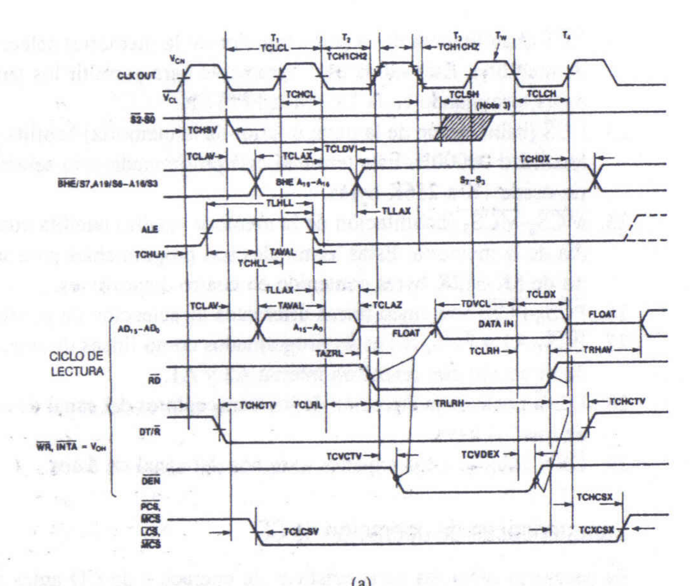
La temporización básica para el 80186/80188 está compuesto de cuatro periodos de reloj como en el 8086/8088. Un ciclo de canal para la versión de 6 MHz requiere 667 ns, mientras que la versión de 16 MHz requiere de 250 ns.

Existen muy pocas diferencias entre la temporización del 80186/80188 y el 8086/8088. La diferencia mas notable es que ALE aparece medio ciclo de reloj entes en el 80186/80188.

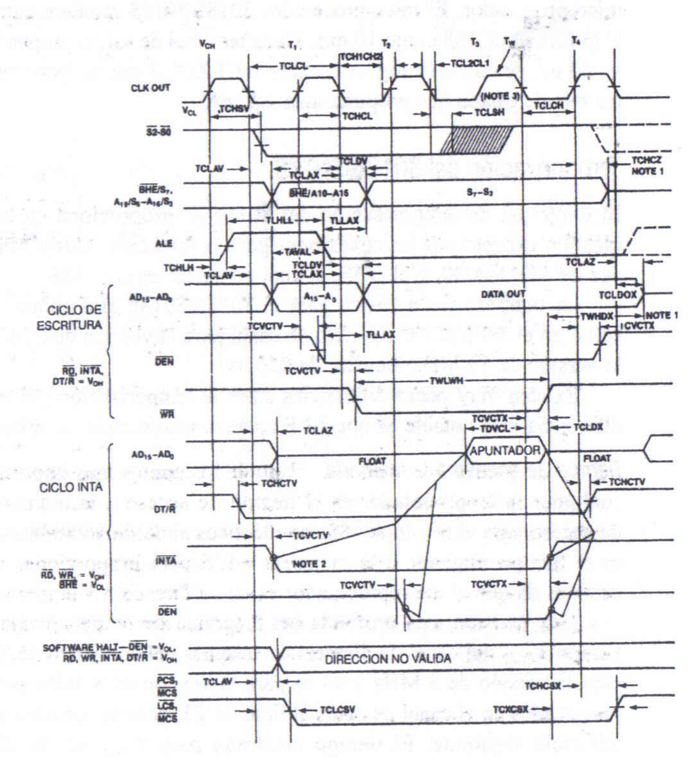
Tiempo de acceso a la memoria. Uno de los puntos más importantes del diagrama de tiempo de cualquier microprocesador es el tiempo de acceso a la memoria. Los cálculos para el tiempo de acceso para el 80186/80188 son idénticos al del 8086/8088. Recuerde que el tiempo de acceso es el tiempo asignado a la memoria y E/S para proporcionar información al microprocesador después de que el microprocesador envía su dirección a la memoria o a E/S.

Una revisión mas profunda del diagrama de tiempo revela que la dirección aparece en el tiempo T_{CLAV} del canal de direcciones después del comienzo de T1. T_{CLAV} está listado como 63 ns para la versión de 6 MHz y 44 ns para la versión de 8 MHz (vea la figura 13-5). Los datos son muesteados en el canal de datos al final de T3, pero se requiere un tiempo de estabilización antes del ciclo siguiente. El tiempo mostrado para T_{DVCL} es de 20 ns para ambas versiones del microprocesador. El tiempo de acceso es por lo tanto igual a tres periodos de reloj menos T_{CLAV} y T_{DVCL}. El tiempo de acceso para el microprocesador a 6 MHz es de 500 ns - 63 ns - 20 ns o 417 ns.

FIGURA 13–4 Diagramas de temporización del 80186/80188. (a) Temporización del ciclo de lectura y (b) temporización del ciclo de escritura. (Cortesía de Intel Corporation.)



(a)



(b)

80186 Master Interface Timing Responses

		80188 (8 MHz)		80188-6 (6 MHz)			
Symbol	**Parameters**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**	**Test Conditions**
T_{CLAV}	Address Valid Delay	5	44	5	63	ns	C_L = 20-200 pF all outputs
T_{CLAX}	Address Hold	10		10		ns	
T_{CLAZ}	Address Float Delay	T_{CLAX}	35	T_{CLAX}	44	ns	
T_{CHCZ}	Command Lines Float Delay		45		56	ns	
T_{CHCV}	Command Lines Valid Delay (after float)		55		76	ns	
T_{LHLL}	ALE Width	$T_{CLCL-35}$		$T_{CLCL-35}$		ns	
T_{CHLH}	ALE Active Delay		35		44	ns	
T_{CHLL}	ALE Inactive Delay		35		44	ns	
T_{LLAX}	Address Hold to ALE Inactive	$T_{CHCL-25}$		$T_{CHCL-30}$		ns	
T_{CLDV}	Data Valid Delay	10	44	10	55	ns	
T_{CLDOX}	Data Hold Time	10		10		ns	
T_{WHDX}	Data Hold after WR	$T_{CLCL-40}$		$T_{CLCL-50}$		ns	
T_{CVCTV}	Control Active Delay 1	5	70	5	87	ns	
T_{CHCTV}	Control Active Delay 2	10	55	10	76	ns	
T_{CVCTX}	Control Inactive Delay	5	55	5	76	ns	
T_{CVDEX}	$\overline{DEN}$ Inactive Delay (Non-Write Cycle)		70		87	ns	
T_{AZRL}	Address Float to $\overline{RD}$ Active	0		0		ns	
T_{CLRL}	$\overline{RD}$ Active Delay	10	70	10	87	ns	
T_{CLRH}	$\overline{RD}$ Inactive Delay	10	55	10	76	ns	
T_{RHAV}	$\overline{RD}$ Inactive to Address Active	$T_{CLCL-40}$		$T_{CLCL-50}$		ns	
T_{CLHAV}	HLDA Valid Delay	10	50	10	67	ns	
T_{RLRH}	$\overline{RD}$ Width	$2T_{CLCL-50}$		$2T_{CLCL-50}$		ns	
T_{WLWH}	$\overline{WR}$ Width	$2T_{CLCL-40}$		$2T_{CLCL-40}$		ns	
T_{AVAL}	Address Valid to ALE Low	$T_{CLCH-25}$		$T_{CLCH-45}$		ns	
T_{CHSV}	Status Active Delay	10	55	10	76	ns	
T_{CLSH}	Status Inactive Delay	10	55	10	76	ns	
T_{CLTMV}	Timer Output Delay		60		75	ns	100 pF max
T_{CLRO}	Reset Delay		60		75	ns	
T_{CHQSV}	Queue Status Delay		35		44	ns	

80186 Chip-Select Timing Responses

Symbol	**Parameter**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**	**Test Conditions**
T_{CLCSV}	Chip-Select Active Delay		66		80	ns	
T_{CXCSX}	Chip-Selct Hold from Command Inactive	35		35		ns	
T_{CHCSX}	Chip-Select Inactive Delay	5	35	5	47	ns	

Symbol	**Parameter**	**Min.**	**Max.**	**Units**	**Test Conditions**
TDVCL	Data in Setup (A/D)	20		ns	
TCLDX	Data in Hold (A/D)	10		ns	
TARYHCH	Asynchronous Ready (AREADY) active setup time*	20		ns	
TARYLCL	AREADY inactive setup time	35		ns	
TCHARYX	AREADY hold time	15		ns	
TSRYCL	Synchronous Ready (SREADY) transition setup time	35		ns	
TCLSRY	SREADY transition hold time	15		ns	
THVCL	HOLD Setup*	25		ns	
TINVCH	INTR, NMI, $\overline{TEST}$, TIMERIN, Setup*	25		ns	
TINVCL	DRQ0, DRQ1, Setup*	25		ns	

*To guarantee recognition at next clock

FIGURA 13–5 Características de AC del 80186. (Cortesía de Intel Corporation.)

13–2 MEJORAS DE PROGRAMACIÓN DEL 80186/80188

Esta sección proporciona detalles sobre la programación y operación de las mejoras del 80186/80188. La siguiente sección detalla el uso del 80186/80188 en un sistema que utiliza muchas de las mejoras que aquí se discuten. La única característica nueva que no queda explicada es el generador del reloj que está completamente descrito en la sección anterior sobre arquitectura.

Bloque de control de periféricos

Todos los periféricos internos son controlados por un conjunto de registros ubicado en el bloque de control de periféricos (PCB). El PCB (vea la figura 13-6) es un conjunto de 256 registros ubicados en el E/S o espacio de la memoria.

Siempre que el 80186/80188 se reinicializa, el bloque de control de periféricos será automáticamente colocado en la parte superior del mapa de E/S (direcciones E/S FF00H-FFFFH). El

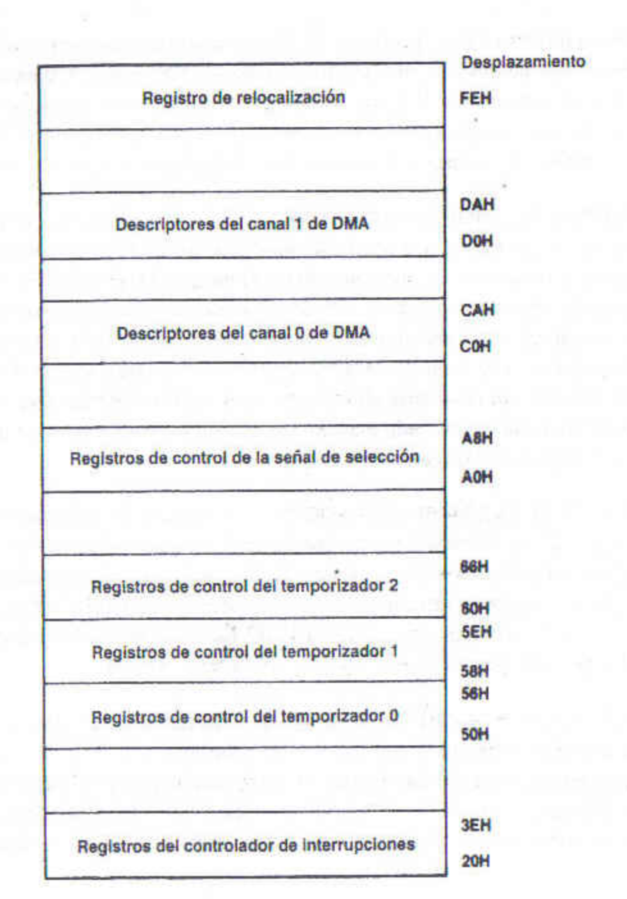


FIGURA 13–6 Bloque de control de periféricos (PCB) del 80186/80188. (Cortesía de Intel Corporation.)

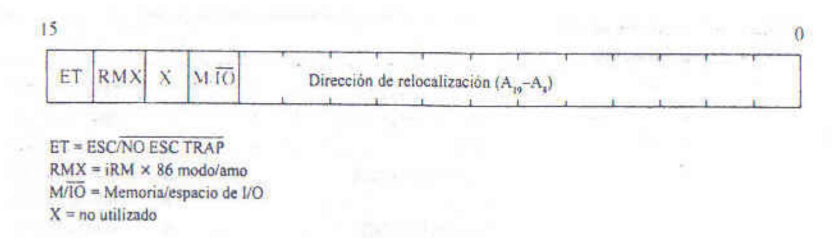


FIGURA 13–7 Registro de relocalización del bloque de control de periféricos.

PBC puede ser reubicado en cualquier momento a cualquier otra área de la memoria o E/S. La relocalización se logra cambiando el contenido del registro de relocalización (vea la figura 13-7) localizado en las direcciones con desplazamientos FEH y FFH.

El registro de relocalización contiene 20FFH cuando se reinicializa el 80186/80188. Esto ubica al PCB en las direcciones FF00H-FFFFH de E/S. Para relocalizar el PCB, el usuario sólo necesita una instrucción OUT a la dirección FFFEH de E/S con un nuevo patrón de bits. Por ejemplo, para relocalizar el PCB a las localidades de memoria 20000H-200FFH, un 1200H se envía a la dirección FFFEH. Observe que M/$\overline{IO}$ es un 1 lógico para selección de memoria y que un 200H selecciona la dirección de memoria 20000H como la dirección base del PCB.

Las interrupciones en el 80186/80188

Las interrupciones en el 80186/80188 son idénticas al 8086/8088, excepto que existen vectores de interrupción adicionales definidos para algunos de los periféricos integrados. Una lista completa de los vectores de interrupción reservados aparece en la tabla 13-1. Los primeros cinco son idénticos a los del 8086/8088.

La interrupción para la instrucción BOUND será requerida si el límite de un arreglo, el cual se inicializa en un registro índice está fuera del valor inicial almacenado en la memoria.

La interrupción para el código de instrucción desconocido ocurrirá cuando el 80186/80188 ejecuta cualquier código de instrucción no definido. Esto es importante si se pierde el control de la ejecución del programa.

La interrupción para el código de la instrucción ESC ocurre si sus códigos D8H-DFH se ejecutan. Esto sólo ocurre cuando el bit ET (trampa) del registro para relocalización está en 1. Si ocurre una interrupción de ESC, la dirección salvada en la pila de memoria por el interruptor señala la instrucción ESC o al prefijo de cambio de segmento si se está usando.

Las interrupciones de los periféricos integrados se deben habilitar mediante la bandera I y se deben desenmascarar para que funcionen. El bit de la bandera I se activa (habilitación) con STI y se desactiva (deshabilitación) con CLI. Las restantes interrupciones internamente decodificadas serán discutidos con los temporizadores y el controlador de DMA más adelante en esta sección.

Controlador de interrupciones

El controlador de interrupciones integrado del 80186/80188 es muy sofisticado. Tiene muchas entradas de interrupción que llegan de las cinco entradas de interrupciones externas, el controlador

TABLA 13–1 Vectores de interrupción del 80186/80188

Nombre	*Tipo*	*Dirección*	*Prioridad*
Error de división	0	00000-00003	1
Paso-sencillo	1	00004-00007	1A
NMI	2	00008-0000B	1
Punto de ruptura	3	0000C-0000F	1
INTO	4	00010-00013	1
Arreglo BOUND	5	00014-00017	1
Código de instrucción no definido	6	00018-0001B	1
ESC código de instrucción ESC	7	0001C-0001F	1
Temporizador 0	8	00020-00023	2A
Temporizador 1	18	00048-0004B	2B
Temporizador 2	19	0004C-0004F	2C
Reservado	9	00024-00027	3
DMA 0	10	00028-0002C	4
DMA 1	11	0002C-0002F	5
INT0	12	00030-00033	6
INT1	13	00034-00037	7
INT2	14	00038-0003B	8
INT3	15	0003C-0003F	9
80187	16	00040-00043	1

Nota: El nivel 1 de prioridad de interrupción es el más alto y 9 es el más bajo. Algunas interrupciones tienen la misma prioridad.

de DMA, y los tres temporizadores. La figura 13-8 proporciona un diagrama a bloques de la estructura del controlador de interrupciones del 80186/80188.

El controlador de interrupciones opera en dos modos: amo e iRMX86[1] o esclavo. El modo es seleccionado en el registro de control de periféricos por el bit RMX. Si RMX es un 1 lógico, el controlador de interrupciones se conecta a controladores de interrupciones externos programables, y si RMX es un 0 lógico, se selecciona el controlador de interrupciones interno.

Esta parte del texto no detalla cómo programar el controlador de interrupciones. En vez de ello se limitará a una discusión de su estructura interna. La programación y aplicación del controlador de interrupciones se discuten en las secciones que describen el temporizador y el controlador de DMA.

Registros del controlador de interrupciones. La figura 13-9 ilustra los registros del controlador de interrupciones, los cuales están ubicados en el bloque de control periféricos que comienzan en la dirección con desplazamiento 20H. Observe que existen dos conjuntos de registros completamente diferentes: uno para el modo de amo interno y otro para el modo (8259A) de esclavo externo iRMX86.

Modo de esclavo iRMX86. Cuando el controlador de interrupciones trabaja en el modo esclavo, utiliza hasta dos controladores de interrupciones externos 8259A, para la expansión de las entra-

[1]iRMX86 es una marca registrada de Intel Corporation.

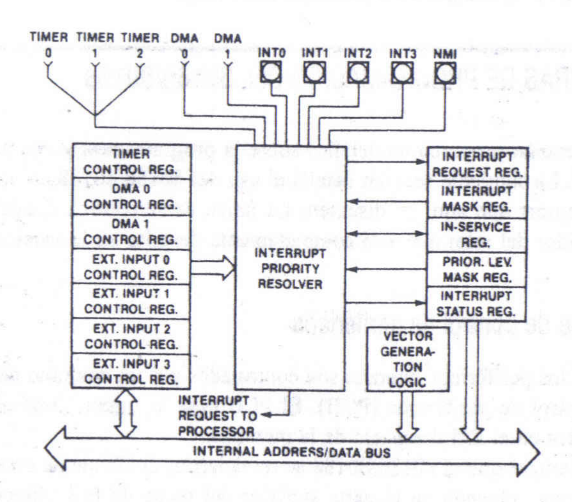


FIGURA 13–8 Controlador programable de interrupciones del 80186/80188. (Cortesía de Intel Corporation.)

das de interrupciones. La figura 13-10 muestra cómo los controladores externos se conectan a las terminales de entrada para interrupciones del 80186/80188 para la operación en modo esclavo. Aquí las entradas de INTO e INTI son utilizadas como conexiones externas para las salidas de solicitud de interrupción de los 8259, e $\overline{INTA}_0(INT_2/\overline{TA}_0)$ e $\overline{INTA}_1(INT_3/\overline{TA}_1)$ son usadas como las señales de reconocimiento para las interrupciones de los controladores externos.

Registros de control de interrupciones. Existen registros de control de interruptores para ambos modos de operación que controlan una sola fuente de interrupción. El modo amo contiene siete registros de control de interrupciones y el modo esclavo iRMX86 contiene cinco. La figura 13-11 describe el patrón binario de bits de cada uno de estos registros de control de las interrupciones. El bit de enmascaramiento habilita (0) o deshabilita (1) la entrada de interrupción representada por la palabra de control, y los bits de prioridades establecen el nivel de prioridad de la fuente de interrupción. El nivel más alto de prioridad es 000 y el más bajo es 111. Los tres bits que sobran se utilizan sólo en el modo amo para seleccionar varios modos de operación que son semejantes a los modos encontrados en el 8259A.

Registro de solicitud de interrupciones. El registro de solicitud de interrupciones contiene una imagen de las fuentes de interrupción en cada uno de los modos de operación. Siempre que se solicite una interrupción, el bit de solicitud de la interrupción correspondiente se convierte en un 1 lógico aunque la interrupción esté enmascarada. La solicitud se borra cuando el 80186/80188 reconoce la interrupción. La figura 13-12 ilustra el patrón binario de bits del registro de solicitud de interrupciones en el modo amo y en esclavo iRMX86.

Registros de enmascaramiento y prioridad. El registro de enmascaramiento de interrupciones tiene el mismo formato que el registro de interrupciones ilustrado en la figura 13-12. Si una fuente esta enmascarada (deshabilitada), el bit correspondiente del registro de enmascaramiento de interrupciones contiene un 1 lógico, y si está habilitado contiene un 0 lógico. El registro de enmascaramiento de interrupciones se lee para determinar cuáles fuentes de interrupción están

Desplazamiento	Registros para el modo amo	Registros para el modo iRMX86 de esclavo
3EH	Registro del control de la INT3	
3CH	Registro del control de la INT2	
3AH	Registro del control de la INT1	Registro del control del nivel 5 (Temporizador 2)
38H	Registro del control de la INT0	Registro del control del nivel 4 (Temporizador 1)
36H	Registro del control de DMA 1	Registro del control del nivel 3 (DMA 1)
34H	Registro de control de DMA 0	Registro del control del nivel 2 (DMA 0)
32H	Registro de control del temporizador	Registro del control del nivel 0 (Temporizador 0)
30H	Registro de estado de las interrupciones	Registro del estado de interruptor
2EH	Registro de solicitud de interrupciones	Registro de solicitud de interrupciones
2CH	Registro de servicio	Registro de en servicio
2AH	Registro de prioridad de enmascaramiento	Registro de prioridad de enmascaramiento
28H	Registro de enmascaramiento	Registro de enmascaramiento
26H	Registro de estado de encuesta	
24H	Registro de encuesta	
22H	Registro EOI	Registro EOI
20H		Registro del vector de interrupción

FIGURA 13–9 Registros para el controlador programable de interrupciones del 80186/80188.

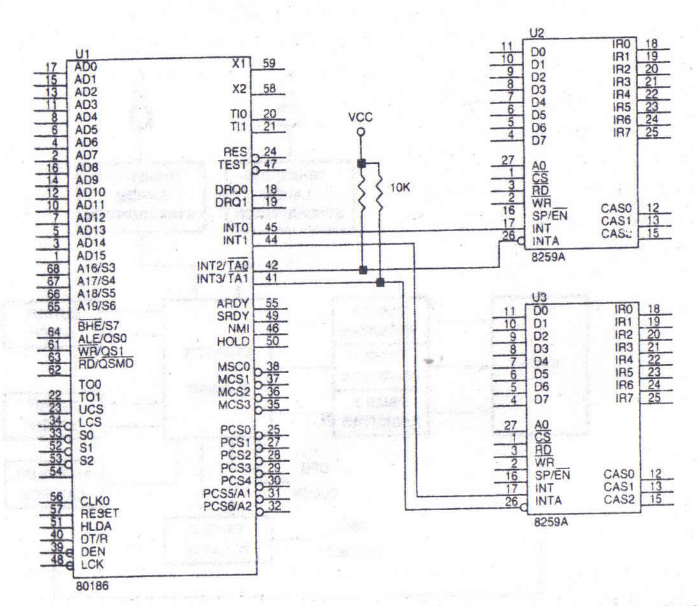
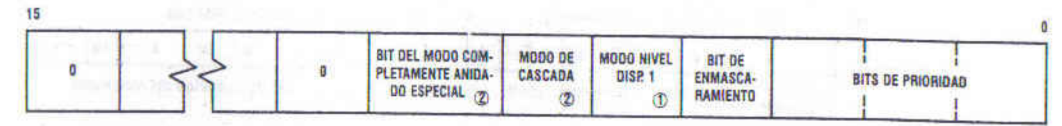


FIGURA 13–10 Dos controladores de interrupciones 8259A conectados al microprocesador 80186. Observe que sólo las conexiones del controlador de interrupciones están ilustradas.



1. Este bit está presente sólo en los en los registros de control into-int3
2. Estos bits se presentan sólo en el registro para control into-int1

FIGURA 13–11 Registro de control de interrupciones. (Cortesía de Intel Corporation.)

MODO DE AMO (15 … 0)

X	X	X	I3	I2	I1	I0	D1	D0	X	TMR

MODO DE IRMX™ 86 (15 … 0)

X	X	X	TMR 2	TMR 1	D0	D1	X	TMR 0

FIGURA 13–12 El registro para la solicitud de interrupciones mostrado para ambos modos de operación. (Cortesía de Intel Corporation.)

enmascaradas y cuáles están habilitadas. Una fuente se enmascara haciendo 1 el bit de enmascaramiento de la fuente de interrupción en el registro de control de la interrupción.

El registro para definir prioridad, ilustrado en la figura 13-13, muestra la prioridad de la interrupción que actualmente atiende el 80186/80188. El nivel de la interrupción lo indican por los bits de prioridad P2-P0. Internamente, estos bits evitarán una interrupción de una prioridad más baja. Estos bits son llevados automáticamente al siguiente nivel más bajo al término de la interrupción que proporciona el 80186/80188. Si ningún otro interruptor está pendiente, estos bits serán activados (111) para habilitar todos los niveles de prioridad.

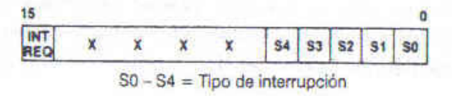
Registro en servicio. El registro de en servicio tiene el mismo patrón de bits binario del registro de solicitud en la figura 13-12. El bit que corresponde a la fuente de interrupción está activo (1) si el 80186/80188 está actualmente reconociendo la interrupción. El bit será desactivado (0) nuevamente al final de la interrupción.

Registros de encuesta y estado de encuesta. Ambos registros tienen el mismo patrón binario de bits como los que se ilustran en la figura 13-14. Estos registros tienen un bit (INT REQ) que indica que una interrupción está pendiente. Este bit se activará si se recibe una interrupción de prioridad suficiente y se desactiva cuando la interrupción sea reconocida. Los bits S indican el número del tipo del vector de interrupción pendiente con la más alta prioridad.

FIGURA 13–13 Registro de prioridad de interrupciones. (Cortesía de Intel Corporation.)



FIGURA 13–14 Registros de encuesta y de estado de encuesta. (Cortesía de Intel Corporation.)



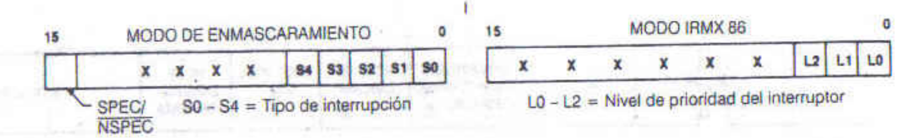


FIGURA 13–15 Registro de fin de interrupción mostrado para ambos modos de operación. (Cortesía de Intel Corporation.)

Estos dos registros pueden aparentar ser idénticos porque contienen la misma información; sin embargo, difieren en su función. Cuando se lee el registro de encuesta de las interrupciones, se reconoce la interrupción. Cuando el registro para estado de encuesta sea leído, ningún reconocimiento se enviará. Estos registros se utilizan sólo en el modo de amo y no en el modo iRMX86 de esclavo.

Registro de fin de interrupción. El registro de fin de interrupción (EOI) ocasiona el fin de una interrupción cuando es escrito por un programa. La figura 13-15 muestra el contenido del registro EOI para el modo de amo y el modo iRMX86 esclavo.

En el modo amo, escribir al registro EOI terminará con una inrterrupción de un nivel específico o de cualquier nivel que esté activo (no específico). En el modo no específico, el bit SPEC/NSPEC debe estar activo antes de que el registro EOI sea escrito para que termine una interrupción no específica.

El EOI no específico borrará el bit de la interrupción del nivel más alto en el registro de en servicio. El EOI específico eliminará el bit seleccionado en el registro de en servicio.

En el no iRMX86 esclavo, el nivel de la interrupción que debe terminar, se escribirá en el registro EOI. El modo de esclavo no permite un EOI no específico.

Registro de estado de las interrupciones. El formato del registro del estado de las interrupciones se describe en la figura 13-16. En el modo de amo, T2-T0 indica cuál temporizador (0, 1, o 2) está causando una interrupción. Esto es necesario porque los tres temporizadores tienen interrupciones del mismo nivel de prioridad. Estos bits se activan cuando un temporizador requiere una interrupción y serán desactivados cuando se reconoce la interrupción. El bit de DHLT (para DMA) sólo se usa en el modo de amo y cuando se activa, detiene una acción de DMA.

Registro para el vector de interrupción. El registro para el vector de interrupción está presente sólo en el modo iRMX86 esclavo y se utiliza para especificar los cinco bits más significativos del número del vector de interrupción. La figura 13-17 ilustra el formato de este registro. Los tres bits menos significativos del número de vector los determina el nivel de prioridad de la interrupción.

Temporizadores

El 80186/80188 contiene tres temporizadores de 16 bits totalmente programables. Cada uno es totalmente independiente de los otros. Dos de los temporizadores de tiempo (0 y 1) tienen terminales de entrada y salida que les permite contar los eventos externos o generar formas de onda. El tercer medidor de tiempo (el No. 2) se conecta al reloj del 80186/80188 y se utiliza como

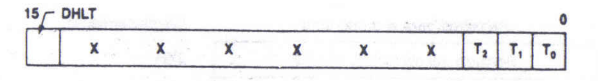


FIGURA 13–16 Registro de estado de interrupciones. (Cortesía de Intel Corporation.)

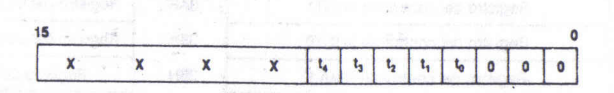


FIGURA 13–17 Registro para vector de interrupción. (Cortesía de Intel Corporation.)

una fuente de solicitud de DMA, una preescala para otros temporizadores, o como vigilante (watchdog).

La figura 13-18 muestra la estructura interna de la unidad de temporización. Observe que la unidad de temporización contiene un elemento contador que es responsable de actualizar los tres contadores. Cada temporizador es realmente un registro que reescribe el contardor (un circuito que lee un valor del temporizador y lo incrementa antes de devolverlo). El elemento contador también es responsable de generar salidas de las terminales T_0OUT y T_1OUT, leer las terminales T_0IN y T_1IN, y ocasionar una solicitud de ciclos de DMA en la cuenta terminal (TC) del temporizador 2, está programado para solicitar ciclos de DMA.

Operación del temporizador. Los temporizadores están controlados por varios registros en el bloque de control de periféricos (vea la figura 13-19). Cada temporizador tiene un registro contador, registro o registros de máxima cuenta, y un registro de control. Estos registros pueden ser todos

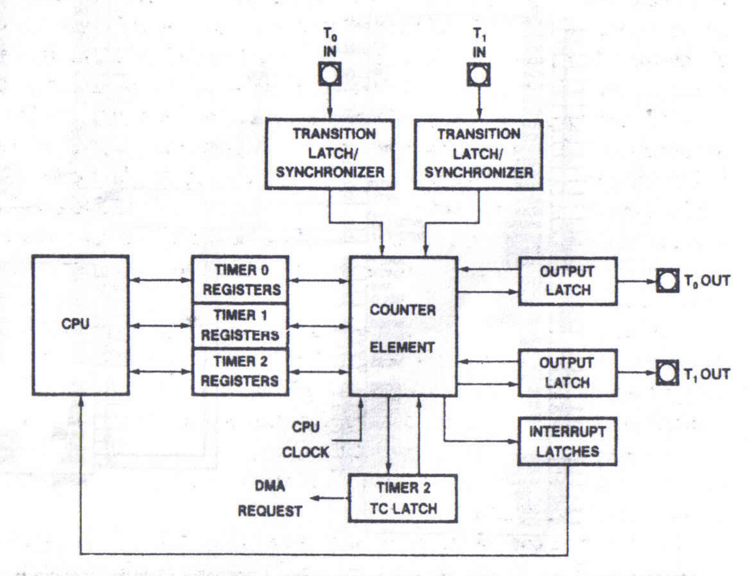


FIGURA 13–18 Estructura interna de los temporizadores del 80186/80188. (Cortesía de Intel Corporation.)

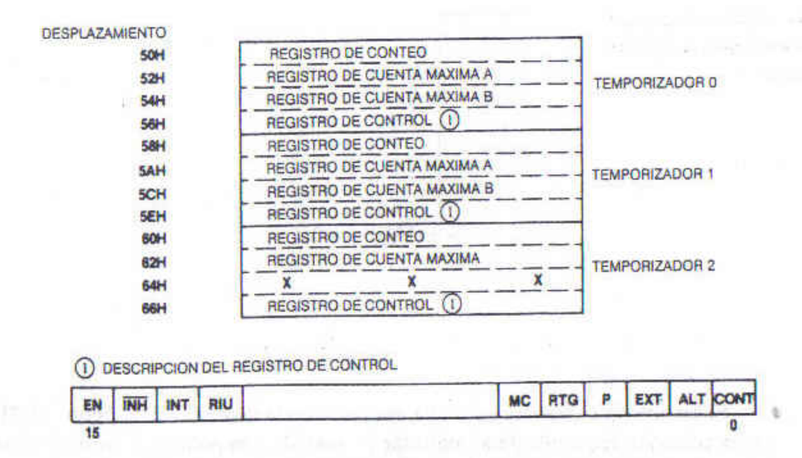


FIGURA 13–19 Registros de los temporizadores. (Cortesía de Intel Corporation.)

leídos o escritos en cualquier momento porque el 80186/80188 se encarga que el contenido nunca cambie durante una lectura o escritura.

El registro contador contiene un número de 16 bits que se incrementa cada vez que hay una entrada al temporizador. Los temporizadores 0 y 1 se incrementan con el flanco positivo de una señal externa aplicada en sus terminales de entrada, a un cuarto de la frecuencia de reloj del 80186/80188, o por la salida del temporizador 2. El temporizador 2 funciona a un cuarto de la frecuencia del reloj del 80186/80188 y no tiene ninguna otra fuente de temporización. Esto significa que en la versión de 8 MHz del 80186/80188, el temporizador 2 opera a 2 MHz y que la frecuencia máxima de conteo de los temporizadores 0 y 1 es de 2 MHz. La figura 13-20 muestra estos cuatro periodos del reloj, los cuales no están relacionados a la temporización canal.

Cada temporizador tiene por lo menos un registro de cuenta máxima (registro A para los temporizadores 0 y 1) que está controlado por el conteo del registro contador para generar una salida. Siempre que el registro contador sea igual al registro de cuenta máxima, se borra a 0. A partir de 0000H, el temporizador tiene una cuenta máxima de 65,536 eventos. Para cualquier otro valor, el temporizador cuenta el valor real del conteo. Por ejemplo, si la cuenta máxima es 0002H, entonces el contador contará de 0 a 1 y después será borrado a 0, un contador de módulo 2 que tiene 2 estados.

Los temporizadores 0 y 1 tienen ambos un segundo registro de cuenta máxima (el registro B) que se selecciona por el registro de control del temporizador. Estos temporizadores utilizan tanto el registro A de cuenta máxima o ambos A y B según se programen con el bit ALT del registro de control del temporizador. Cuando ambos registros de cuenta máxima se utilizan, el temporizador cuenta hasta el valor del registro A, lo borra a 0, y después cuenta la cantidad del registro B de cuenta máxima. Este proceso se repite. El utilizar ambos registros de cuenta máxima le permite al temporizador contar hasta 131,072.

El registro de control (refiérase de nuevo a la figura 13-19) de cada temporizador mediante 10 bits especifica la operación del temporizador. A continuación se encuentra una definición de cada bit de control:

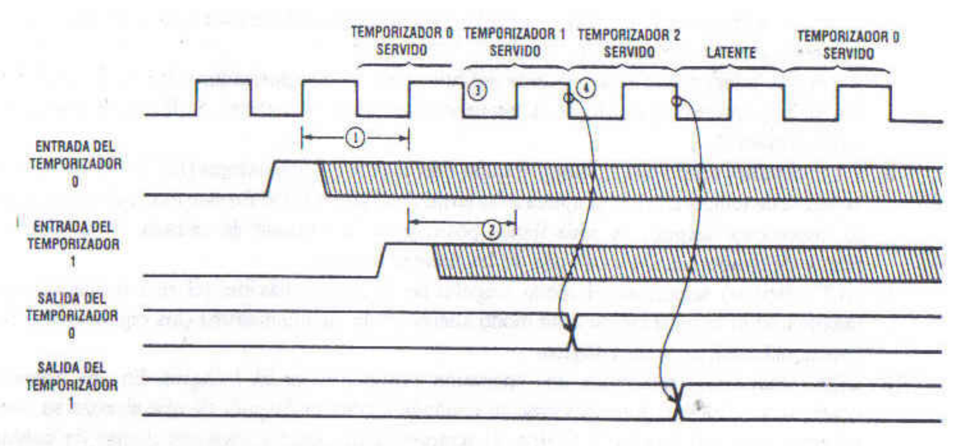


FIGURA 13-20 Diagrama de temporización de los temporizadores del 80186/80188. (Cortesía de Intel Corporation.)

1. EN (habilitación) permite al temporizador empezar a contar. Si EN se desactica (O), el temporizador no contará. Si se activa, el temporizador cuenta.
2. INH (inhibir) permite que la escritura al registro de control del temporizador afecte al bit de habilitación (EN). Si INH es uno, entonces el bit EN se puede activar o desactivar para controlar el conteo. Si INH es cero elimina, EN no será afectado por la escritura al registro de control del temporizador. Esto permite que otras características del medidor de tiempo sean modificadas sin habilitar o deshabilitar el temporizador.
3. INT (interrupción) permite que el temporizador genere una interrupción. Si INT está activo (1), ocurrirá una interrupción cada vez que la cuenta máxima sea alcanzada en cualquiera de los registros de máxima cuenta. Si el bit se desactiva, no se genera ninguna interrupción. Cuando se genera una solicitud de interrupción se genera, permanece activa aun si el bit EN se desactica después de la solicitud de interrupción.
4. RIU (registro en uso) indica cuál registro de cuenta máxima está usando actualmente el temporizador. Si RIU es un 0 lógico, entonces el registro de cuenta máxima A está en uso. Este bit es un bit sólo de lectura, y escribir en él no le afectará.
5. MC (cuenta máxima) indica que el temporizador ha alcanzado su cuenta máxima. Este bit se convierte en 1 lógico cuando el temporizador alcanza su cuenta máxima y permanece como 1 lógico hasta que el bit MC se borra escribiendo un 0 lógico. Esto permite que la cuenta máxima sea detectada por un programa.
6. RTG (disparo) es activo sólo para relojes externos (EXT = 0). El bit RTG se utiliza sólo con los temporizadores 0 y 1 para seleccionar la operación de las teminales de entrada del temporizador (T_0IN y T_1IN). Si RTG es un 0 lógico, la entrada externa causará que el temporizador cuente si es un 1 lógico y mantener la cuenta (dejar de contar) si es un 0 lógico. Si RTG es un

1 lógico, la terminal de entrada externa borrará la cuenta del temporizador a 0000H cada vez que ocurra un flanco positivo.

7. P (preescalador) selecciona la fuente del reloj para los temporizadores 0 y 1. Si EXT = 0 y P = 0, la fuente es un cuarto de la frecuencia del reloj del sistema. Si P = 1, la fuente es el temporizador 2.
8. EXT (externo) selecciona la temporización interna (EXT = 0) o externa (EXT = 1). Si EXT = 1, la fuente de temporización se aplica a las terminles T_0IN o T_1IN. En este modo el temporizador se incrementa después de cada flanco positivo en la terminal de entrada. Si EXT = 0, la fuente de temporización es de una de las fuentes internas.
9. ALT (alternar) selecciona el modo singular de la cuenta máxima (el registro de la cuenta máxima A) si es un 0 lógico o un modo alterno de la cuenta máxima (los registros A y B de cuenta máxima) si es un 1 lógico.
10. CONT (continuo) selecciona una operación continua si es un 1 lógico. En una operación continua, el contador automáticamente continúa contando después de que alcanza su cuenta máxima. Si CONT es un 0 lógico, el temporizador automáticamente dejará de contar y desactivará al bit EN. Observe que cuando el 80186/80188 se reinicializa, los temporizadores son deshabilitados automáticamente.

Terminal de salida del temporizador. Los temporizadores 1 y 0 tienen una terminal de salida utilizada para generar ya sean ondas cuadradas o pulsos. Para producir pulsos, el temporizador se opera en el modo sencillo de cuenta máxima (ALT = 0). En este modo, la termial de salida va a un nivel bajo por un periodo de reloj cuando el contador alcanza su cuenta máxima. Controlando el bit CONT en el registro de control, se pueden generar uno o varios pulsos.

Para producir ondas cuadradas o con ciclos de trabajo variables, se selecciona el modo alterno (ALT = 1). En este modo, la terminal de salida es un 1 lógico mientras que el registro A de cuenta máxima controla al temporizador y 0 lógico mientras el registro B de cuenta máxima B controla al temporizador. Igual que con el modo sencillo de cuenta máxima, el temporizador puede generar ya sea una sola onda cuadrada u ondas cuadradas en forma continua. Refiérase a la tabla 13-2 para el funcionamiento de los bits de control ALT y CONT.

Casi cualquier ciclo de trabajo puede generarse en el modo alterno. Por ejemplo, suponga que un ciclo de trabajo del 10 por ciento se requiere en la terminal de salida de un temporizador. El registro de cuenta máxima A se cargará con un 10 y el registro de la cuenta máxima B con 90, para producir una salida que sea un 1 lógico durante 10 ciclos de reloj y un 0 lógico durante 90 ciclos de reloj. Esto también divide la frecuencia de la fuente de temporización por un factor de 100.

Ejemplo de un reloj de tiempo real. Muchos sistemas requieren la hora del día. Esto frecuentemente se llama un reloj de tiempo real. Uno de los temporizadores del 80186/80188 puede proporcionar la fuente de temporización para el programa que lleva la hora del día.

TABLA 13–2 Función de ALT y CONT en el registro de control del temporizador

ALT	*CONT*	*Modo*
0	0	un solo pulso
0	1	pulsos continuos
1	0	una sola onda cuadrada
1	1	ondas cuadradas continuas

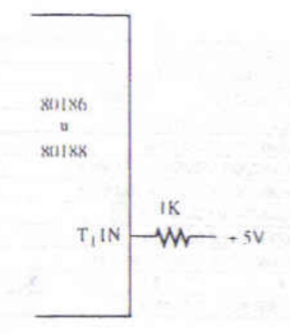


FIGURA 13-21 Circuito para el temporizador para un reloj de tiempo real.

El circuito que requiere para esta aplicación está ilustrado en la figura 13-21. Observe que la única conexión requerida para implantar un reloj de tiempo real es un solo resistor conectado de la terminal T_1IN a +5.0 V para habilitar al temporizador 1. En el ejemplo, los temporizadores 1 y 2 son utilizados para generar una interrupción de cada segundo que proporciona al programa una fuente de temporización.

El programa requerido para implantar un reloj de tiempo real se lista en el ejemplo 13-1. Estos son los dos procedimientos: uno programa los temporizadores 1 y 2 y el otro, un procedimiento de servicio de interrupción, para llevar el tiempo. Este es el tercer procedimiento que incrementa un contador de módulo BCD. Ni el programa requerido para instalar el vector de interrupción ni el que lleva la hora del día se ilustran.

EJEMPLO 13-1

```
                ;Software para controlar los temporizadores para un reloj de tiempo real.
                ;
                ;Se asume que el bloque de control de periférico (PCB)
                ;está ubicado después de la reinicialización en los puertos de E/S en FF00H--FFFFH.
                ;
= FF62          TIM2_MCOUNT     EQU   0FF62H        ;cuenta máxima del temporizador 2
= FF66          TIM2_CONTR      EQU   0FF66H        ;control del temporizador
= FF5A          TIM1_MCOUNT     EQU   0FF5AH        ;cuenta máxima del temporizador 1
= FF5E          TIM1_CONTR      EQU   0FF5EH        ;control del temporizador

0000            INICIA          PROC  NEAR

0000  B8 07D0                   MOV   AX,2000       ;cuenta de carga del temporizador 2
0003  BA FF62                   MOV   DX,TIM2_MCOUNT
0006  EF                        OUT   DX,AX

0007  B8 C001                   MOV   AX,0C001H     ;habilitar el temporizador 2
000A  BA FF66                   MOV   DX,TIM2_CONTR
000D  EF                        OUT   DX,AX

000E  B8 0064                   MOV   AX,100        ;cargar cuenta del temporizador 1
0011  BA FF5A                   MOV   DX,TIM1_MCOUNT
0014  EF                        OUT   DX,AX

0015  B8 E009                   MOV   AX,0E009H     ;habilitar el temporizador 1
0018  BA FF5E                   MOV   DX,TIM1_CONTR
```

```
001B   EF                        OUT   DX,AX
001C   C3                        RET

001D           INICIA            ENDP


001D           INTERRUPCION      PROC  FAR

001D   56                        PUSH  SI
001E   50                        PUSH  AX

001F   BE 0000 R                 MOV   SI,SEGUNDOS DESPLAZADOS
0022   B4 60                     MOV   AH,60H

0024   E8 0038 R                 CALL  UP_COUNT        ;incrementar segundos
0027   75 0C                     JNZ   ENDI
0029   46                        INC   SI

002A   E8 0038 R                 CALL  UP_COUNT        ;incrementar minutos
002D   75 06                     JNZ   ENDI
002F   46                        INC   SI
0030   B4 24                     MOV   AH,24H
0032   E8 0038 R                 CALL  UP_COUNT        ;incrementar horas

0035                       ENDI:

0035   58                        POP   AX
0036   5E                        POP   SI
0037   CF                        IRET

0038               INTERRUPCION EDNP

0038               CUENTA_SUP PROC  NEAR
0038 8A 04                    MOV   AL,[SI]
003A 04 01                    ADD   AL,1
003C 27                       DAA
003D 88 04                    MOV   [SI],AL
003F 2A C4                    SUB   AL,AH
0041 75 02                    JNZ   FIN_SUP
0043 88 04                    MOV   [SI],AL

0045                          FIN_SUP:

0045 C3                       RET

0046               CUENTA_SUP ENDP
```

El temporizador 3 esta programado para dividir por un factor de 20000. Esto ocasiona que el reloj (2 MHz en la versión 8 MHz del 80186/80188) sea dividido para producir un pulso cada 10 ms. El reloj para el temporizador 1 se deriva de la salida del temporizador 2. El temporizador 1 está programado para dividir por 100 y generará un pulso por segundo. El registro para control del temporizador 1 está programado para que genere una interrupción una vez por segundo.

El procedimiento del servicio de la interrupción se llama una vez por segundo para llevar el tiempo. Este procedimiento incrementa el contenido de la localidad de la memoria SEGUNDOS. Una vez cada 60 segundos, el contenido de la siguiente localidad de la memoria (SEGUNDOS + 1) se incrementa. Finalmente, una vez por hora, el contenido de la localidad de la memoria SEGUNDOS + 2 se incrementa. El tiempo se archiva en estas tres localidades consecu-

tivas de la memoria en BCD para que los programas del sistema puedan fácilmente tener acceso al tiempo.

Controlador de DMA

El controlador de DMA integrado del 80186/80188 tiene dos canales DMA completamente independientes. Cada uno tiene su propio conjunto de registros de direccionamiento de 20 bits así que cualquier localidad de memoria o de E/S es accesible para una transferencia de DMA. Además, cada canal es programable para el autoincremento o decremento de los registros fuente o destino.

La figura 13-22 ilustra la estructura interna del registro del controlador de DMA. Estos registros están ubicados en el bloque de control de periférico en las direcciones con desplazamiento C0H-DFH.

Observe que el conjunto de registros de ambos canales DMA son idénticos. Cada canal contiene una palabra de control, un apuntador fuente y un apuntador destino, y un contador de transferencias. El contador de transferencias es de 16 bits y permite las transferencias no atendidas de DMA de bytes (80188/80186) y palabras (únicamente 80186). Cada vez que un byte o una palabra se transfiere, el contador se decrementa 1 hasta que alcance 0000H, la cuenta terminal.

Los apuntadores fuente y destino son cada uno de 20 bits de ancho, así que las transferencias de DMA pueden ocurrir a cualquier localidad de la memoria o puerto de E/S sin considerar ni el segmento ni el desplazamiento. Si la dirección de la fuente o del destino es un puerto E/S, los bits A19-A16 deben ser 0000 o podrá ocurrir un funcionamiento equivocado.

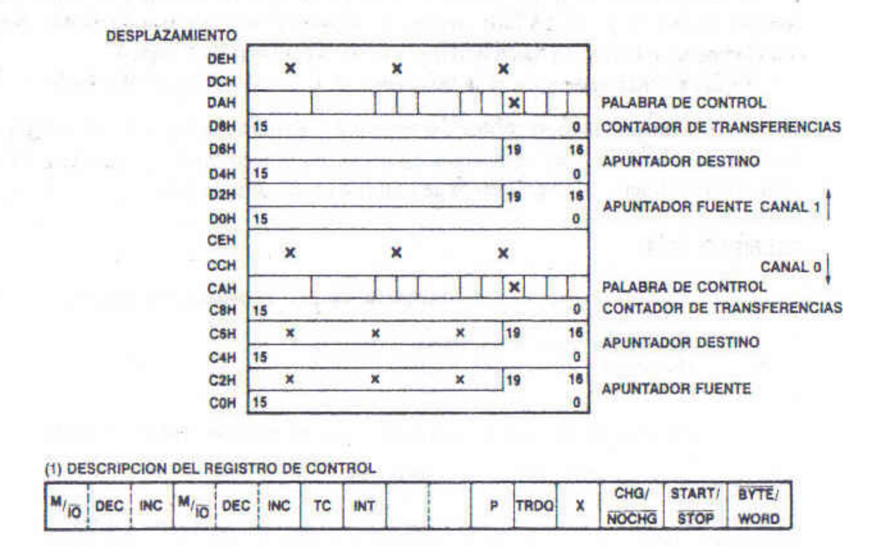


FIGURA 13-22 Estructuras de registros del controlador de DMA del 80186/80188. (Cortesía de Intel Corporation.)

Registro de control del canal. Cada canal de DMA contiene su propio registro de control (refiérase a la figura 13-22), el cual define su operación. Los 6 bits de la izquierda especifican la operación de los registros fuente y destino. El bit M/IO indica una localidad de memoria o de E/S, DEC causa que el apuntador se decremente, e INC ocasiona que el apuntador se incremente. Si los bits INC y DEC son 1, entonces el apuntador permanece sin cambio después de cada transferencia de DMA. Observe que las transferencias de memoria-a-memoria son posibles con este controlador de DMA.

El bit TC (cuenta terminal) hace que el canal de DMA detenga las transferencias cuando el registro para la cuenta de canal se decrementa hasta 0000H. Si este bit es un 0 lógico, el controlador de DMA continúa transfiriendo datos aun después de que la cuenta de la terminal se almacene.

El bit INT habilita las interrupciones en el controlador de interrupciones. Si se activa (1), este bit causa una interrupción cuando se alcance la cuenta terminal del canal.

El bit SYN selecciona el tipo de sincronización para el canal: 00 = no sincronización, 01 = sincronización de la fuente, y 10 = sincronización del destino. Cuando se selecciona sin sincronización o sincronización de fuente, la información será transferida a la velocidad de 2 M bytes por segundo. Estos dos tipos de sincronización permiten que las transferencias ocurran sin interrupción. Si se selecciona la sincronización del destino, la velocidad de transferencia será más lenta (1.3 M bytes por segundo), y el controlador cede el control al 80186/80188 después de cada transferencia de DMA.

El bit P selecciona la prioridad del canal. Si P = 1, el canal tiene la prioridad más alta. Si ambos canales tienen la misma prioridad, el controlador alterna transferencias entre los canales.

El bit TRDQ habilita transferencias de DMA iniciadas por temporizador 2. Si este bit es de un 1 lógico, la solicitud de DMA la origina el temporizador 2. Esto puede prevenir que las transferencias de DMA utilicen todo el tiempo del microprocesador para la transferencia.

El bit CHG/$\overline{\text{NOCHG}}$ determina si el bit START/$\overline{\text{STOP}}$ cambia al escribir al registro para control. El bit de START/$\overline{\text{STOP}}$ arranca o detiene la transferencia de DMA. Para arrancar la transferencia de DMA, CHG/$\overline{\text{NOCHG}}$ y START/$\overline{\text{STOP}}$ son un 1 lógico.

$\overline{\text{BYTE}}$/WORD selecciona si la transferencia es de tamaño palabra o byte.

Muestra para transferencia de memoria-a-memoria. El controlador de DMA integrado es capaz de realizar transferencias de memoria-a-memoria. El procedimiento utilizado para programar el controlador e iniciar la transferencia se muestra en el ejemplo 13-2.

EJEMPLO 13–2

```
                    ;Procedimiento de transferencia de DMA de memoria-a-memoria
                    ;
                    ;Dirección fuente es DS:SI
                    ;Dirección destino es ES:DI
                    ;Cuenta en CX
                    ;
                    ;El bloque de control periférico (PCB) está en FF00H -- FFFFH
                    ;
0000                     MOVE_BYTES     PROC FAR

0000 8C D8                              MOV      AX,DS      ;forma fuente
0002 C1 E0 04                           SHL      AX,4
0005 03 C6                              ADD      AX,SI
0007 BA FFC0                            MOV      DX,0FFC0H
000A EF                                 OUT      DX,AX
000B 9C                                 PUSHF
```

```
000C 8C D8                 MOV      AX,DS
000E C1 E8 0C              SHR      AX,12
0011 9D                    POPF
0012 05 0000               ADD      AX,0
0015 83 C2 02              ADD      DX,2
0018 EF                    OUT      DX,AX

0019 8C C0                 MOV      AX,ES      ;forma destino
001B C1 E0 04              SHL      AX,4
001E 83 C2 02              ADD      DX,2
0021 03 C7                 ADD      AX,DI
0023 EF                    OUT      DX,AX
0024 9C                    PUSHF
0025 8C C0                 MOV      AX,ES
0027 C1 E8 0C              SHR      AX,12
002A 9D                    POPF
002B 05 0000               ADD      AX,0
002E 83 C2 02              ADD      DX,2
0031 EF                    OUT      DX,AX

0032 8B C1                 MOV      AX,CX      ;contador del programa
0034 83 C2 02              ADD      DX,2
0037 EF                    OUT      DX,AX

0038 B8 B606               MOV      AX,0B606H  ;control del programa
003B 83 C2 02              ADD      DX,2
003E EF                    OUT      DX,AX      ;arrancar transferencia

003F CB                    RET

0040           MOV_BYTES   ENDP
```

Este procedimiento transfiere información de la localidad del segmento de datos direccionados por SI a la localidad del segmento extra direccionado por DI. El número de bytes transferidos está en el registro CX. Esta operación es idéntica a la instrucción REP MOVSB, pero la ejecución ocurre a una velocidad mucho mas rápida.

Unidad de señales de habilitación

La unidad de señales de habilitación simplifica la interface de la memoria y E/S del 80186/80188. Esta unidad contiene una lógica programable para señales de habilitación. En los sistemas pequeños y medianos, ningún decodificador externo se requiere para habilitar memoria y E/S. Sin embargo, los sistemas grandes, pueden requerir decodificadores externos.

Señales de habilitación de memoria. Hay disponibles seis señales de habilitación para seleccionar los diferentes componentes de memoria en un sistema de tamaño pequeño o mediano basado en el 80186/80188. La terminal UCS (seleccionar memoria superior) activa la parte de la memoria ubicada en la porción superior del mapa de la memoria que típicamente es ROM. Esta terminal es programable y permite que el tamaño de la ROM se especifique y también el número de estados de espera requeridos. Observe que la dirección final de la ROM es FFFFFH.

La terminal $\overline{LCS}$ (seleccionar memoria inferior) selecciona la parte de la memoria (normalmente RAM) que comienza en la localidad 00000H. Como con $\overline{UCS}$, el tamaño de la memoria y número de los estados de espera son programables.

Las cuatro terminales restantes seleccionan dispositivos en la parte media de la memoria. Estas cuatro terminales ($\overline{MCS3}$-$\overline{MCS0}$) están programadas tanto para la dirección (base) inicial como para el tamaño de la memoria. Observe que todos los dispositivos deben ser del mismo tamaño.

Señales de habilitación de periféricos. El 80186/80188 puede direccionar hasta siete dispositivos periféricos externos con las terminales $\overline{PCS6}$-$\overline{PCS0}$. La dirección de base E/S se programa en cualquier intervalo de 1K byte en bloques de 128 bytes de direcciones de puerto.

Cómo programar la unidad de señales de habilitación. El número de estados de espera en cada sección de la memoria y E/S son programables. El 80186/80188 tiene un generador de estado de espera integrado que puede introducir entre 0 y 3 estados de espera. La tabla 13-3 muestra los niveles lógicos requeridos en los bits R2-R0 en cada registro programable para seleccionar varios números de estados de espera. Estas tres líneas también seleccionan si se requiere una señal externa de READY para generar estados de espera. Si se selecciona READY, la señal externa de READY estará en paralelo con el generador de estados de espera interno. Por ejemplo, si READY es un 0 lógico por tres periodos de reloj, pero el generador de estados de espera interno está programado para insertar 2 estados de espera, serán insertados tres.

Suponga que un EPROM de 64K está ubicada en la parte superior del sistema de la memoria y requiere de 2 estados de espera para funcionar correctamente. Para seleccionar este dispositivo para esta sección de la memoria, la terminal $\overline{UCS}$ se programa para un rango de memoria de F0000H-FFFFFH con 2 estados de espera. La figura 13-23 muestra los registros de control para las señales de habilitación de la memoria y de E/S en el bloque de control de periféricos en las direcciones con desplazamientos A0-A9H. Observe que los 3 bits a la derecha de estos registros de control son los de la tabla 13-3. El registro de control del área superior de la memoria está ubicado en la dirección con desplazamiento A0H en el PCB. Este registro de 16 bits se programa con la dirección inicial del área de la memoria (en este caso F0000H) y el número de estados de espera. Refiérase a la tabla 13-4 para los códigos válidos para varios tamaños de la memoria sin ningún estado de espera. Debido a que nuestro ejemplo requiere de 2 estados de espera, la dirección básica es igual a la de la tabla para un dispositivo de 64K, excepto que los tres bits más a la derecha son 110 en vez de 000. La información enviada al registro de control de la memoria superior son F03EH.

Suponga que un SRAM de 32K que no requiere no ciclos de espera ni entrada READY está ubicada en la parte baja del sistema de memoria. Para programar la terminal $\overline{LCS}$ para que la

TABLA 13-3 Bits de control de los estados de espera R2, R1, y R0

R2	*R1*	*R0*	*Número de estados*	*READY requerido*
0	0	0	0	Sí
0	0	1	1	Sí
0	1	0	2	Sí
0	1	1	3	Sí
1	0	0	0	No
1	0	1	1	No
1	1	0	2	No
1	1	1	3	No

FIGURA 13–23 Estructura del registro de la unidad de señales de habilitación del 80186/80188. (Cortesía de Intel Corporation.)

DESPLAZAMIENTO:

Desplazamiento	Registro				Señal
A0H	TAMAÑO DE LA MEMORIA SUPERIOR			①	UMCS
A2H	TAMAÑO DE LA MEMORIA INFERIOR			②	LMCS
A4H	DIRECCION BASE PARA LA HABILITACION DE PERIFERICOS			③	PACS
A6H	DIRECCION BASE DE MEMORIA MEDIA			④	MMCS
A8H	TAMAÑO DE MEMORIA MEDIA	EX	MS ⑥	⑤	MPCS

1. Bits de espera de la memoria superior
2. Bits de espera de la memoria inferior
3. Bits de espera PCS0-PCS3
4. Bits de espera de la memoria media
5. Bits de espera de PCS4-PCS6
6. MS: 1 = Periféricos activos en el espacio de memoria
   0 = Periféricos activos en el espacio de E/S
   EX:1 = 7 líneas PCS
   0 = PCS5 = A1, PCS6 = A2

No todos los bits de cada campo son utilizados

habilite, el registro en el desplazamiento A2 se carga de exactamente la misma manera que el registro en el A0H. En este ejemplo, se envía 07FCH al registro en el desplazamiento A2H. La tabla 13-5 muestra los valores de programación válidos para la terminal selección de la parte inferior.

La parte media de la memoria se programa por medio de dos registros: con desplazamientos A6H y A8H. El registro A6H programa la dirección base de las líneas de habilitación de la memoria media ($\overline{MCS3}$-$\overline{MCS0}$) y los ciclos de espera. El registro en el desplazamiento A8H define el tamaño del bloque de memoria y el tamaño de los dispositivos de la memoria individual (consulte a la tabla 13-6). Además del tamaño del bloque, el número de estados de espera de los periféricos se programan como otras áreas de la memoria. El bit 7 (EX) y el bit 6 (MS) especifican las líneas de habilitación de periféricos y se discutirán más adelante.

Por ejemplo, suponga que cuatro SRAM de 32K bytes se agregan al área de la memoria media comenzando en la localidad 80000H y terminando en la localidad 9FFFFH sin ningún

TABLA 13–4 Programación de la memoria superior para el registro con desplazamiento A0H

Dirección inicial	*Tamaño del bloque*	*Valor sin esperas, sin READY*
FFC00H	1K	FFF8H
FF800H	2K	FFB8H
FF000H	4K	FF38H
FE000H	8K	FE38H
FC000H	16K	FC38H
F8000H	32K	F838H
F0000H	64K	F038H
E0000H	128K	E038H
C0000H	256K	C038H

TABLA 13–5 Programación de la memoria inferior para el registro con desplazamiento A2H

Dirección final	*Tamaño del bloque*	*Valor sin esperas, sin READY*
003FFH	1K	0038H
007FFH	2K	0078H
00FFFH	4K	00F8H
01FFFH	8K	01F8H
03FFFH	16K	03F8H
07FFFH	32K	07F8H
0FFFFH	64K	0FF8H
1FFFFH	128K	1FF8H
3FFFFH	256K	3FF8H

estado de espera. Para programar las líneas de selección de la memoria media para esta área de la memoria, colocamos los siete bits más a la izquierda de las direcciones en el registro con desplazamiento A6H con bits del 8 al 3 conteniendo unos lógicos y los tres bits más a la derecha conteniendo los bits de control de los estados de espera (READY). Para este ejemplo, el registro con desplazamiento A6H se carga con 81FCH. El registro con desplazamiento A8H está programado con 903CH asumiendo que EX = MS = 0 y ni estado de espera interno ni externo (READY) se requiere para los periféricos.

El registro en el desplazamiento A4H programa a las terminales de selección de periféricos ($\overline{PCS0}$-$\overline{PCS6}$) junto con los bits EX y MS en el registro con desplazamiento A8H. El registro con desplazamiento A4H tiene la dirección inicial base de las líneas de selección para periféricos. Estos periféricos pueden estar mapeados en memoria o en E/S. Si están en el espacio de E/S, los bits de direccionamiento A19-A16 son 0000. Una vez que la dirección de inicio está programada en cualquier espacio de 1K byte, las terminales $\overline{PCS}$ están espaciadas en intervalos de 128 bytes.

Por ejemplo, si el registro en el desplazamiento A4H se programa con 00FCH, sin ningún estado de espera y sin sincronización de la terminal READY, la dirección de la memoria iniciará en 00C00H o los puertos de E/S en 0C00H. En este caso, los puertos E/S son: $\overline{PCS0}$ = 0C00H, $\overline{PCS1}$ = 0C80H, $\overline{PCS2}$ = 0D00H, $\overline{PCS3}$ = 0D80H, $\overline{PCS4}$ = 0E00H, $\overline{PCS5}$ = 0E80H, y $\overline{PCS6}$ = 0F00H.

TABLA 13–6 Programación de la memoria media para el registro con desplazamiento A8H

Tamaño del bloque	*Tamaño del dispositivo*	*Valor sin esperas, sin READY, y EX = MS = 0*
8K	2K	8138H
16K	4K	8238H
32K	8K	8438H
64K	16K	8838H
128K	32K	9038H
256K	64K	A038H
512K	128K	C038H

El bit MS del registro en el desplazamiento A8H selecciona que los periféricos estén mapeados en memoria o en E/S. Si MS es un 1 lógico, entonces las líneas $\overline{PCS}$ se decodifican en el mapa de la memoria. Si es un 0 lógico, entonces las líneas $\overline{PCS}$ están en el mapa de E/S.

El bit EX selecciona la función de las terminales $\overline{PCS5}$ y $\overline{PCS6}$. Si EX = 1, estas terminales de $\overline{PCS}$ seleccionan los periféricos de E/S. Si EX = 0, proporcionan al sistema con las líneas de dirección A1 y A2 con retención. A1 y A2 se usan en algunos equipos de E/S para seleccionar los registros internos y se proporcionan para este propósito.

13–3 EJEMPLO DE UNA INTERFACE PARA 80188

Debido a que el 80186/80188 está diseñado como un controlador dedicado, esta sección del texto proporciona un ejemplo de esa aplicación. El ejemplo ilustra un sistema sencillo con memoria y E/S basado en el microprocesador 80188. También muestra el programa requerido para programar al 80188 y sus registros internos después de la reinicialización del sistema. El programa para controlar el sistema no está incluido.

Puede desarrollarse un sistema, para aprender a utilizar los microprocesadores de la familia 8086/8088, basado en un microprocesador 80188. El ayudante utiliza una EPROM 2764 para el almacenamiento de programas, tres SRAM 62256 para el almacenamiento de datos, un controlador de teclado/exhibición 8279, y un USART 8251A. La figura 13-24 ilustra este pequeño sistema basado en el microprocesador 80188.

La memoria EPROM 2674 la habilita la terminal $\overline{UCS}$ para la $\overline{LCS}$ una de las SRAM, y $\overline{MCS0}$ y $\overline{MCS1}$ seleccionan las SRAM restantes. Los periféricos son habilitados por $\overline{PCS0}$ y $\overline{PCS1}$, $\overline{PCS0}$ habilita al 8279 y $\overline{PCS1}$ habilita al 8251A. Observe que se programan dos estados de espera para la EPROM, el 8251A y 8279 porque requieren de más tiempo de acceso que el SRAM, la que funciona sin ningún estado de espera.

El sistema coloca a EPROM en las direcciones de la memoria FE000H-FFFFFH; la SRAM en 00000H-07FFFH, 80000H-87FFFH, y 88000H-8FFFFH; el 8279 en los puertos de E/S 1000H-107FH; y el 8251A en los puertos E/S 1080H-108FH. En este ejemplo no modificamos la dirección del bloque de control de periféricos, el cual se ubica en los puertos E/S FF00H-FFFFH.

EJEMPLO 13–3

```
                        ;Programa de inicialización para el sistema basado en el microprocesador 80188
FFF0                            ORG     0FFF0H

FFF0 EB 8E                      JMP     INICIA

FF80                            ORG     0FF80H

FF80                    INICIA
FF80 B8 FF3E                    MOV     AX,0FE3EH               ;2 esperas, bloque 8K
FF83 BA FFA0                    MOV     DX,0FFA0H               ;dirección A0H
FF86 EF                         OUT     DX,AX                   ;programar la memoria superior
FF87 B8 07FC                    MOV     AX,07FCH                ;0 esperas, bloque 32K
FF8A 83 C2 02                   ADD     DX,2                    ;dirección A2H
FF8D EF                         OUT     DX,AX                   ;programar la memoria inferior
```

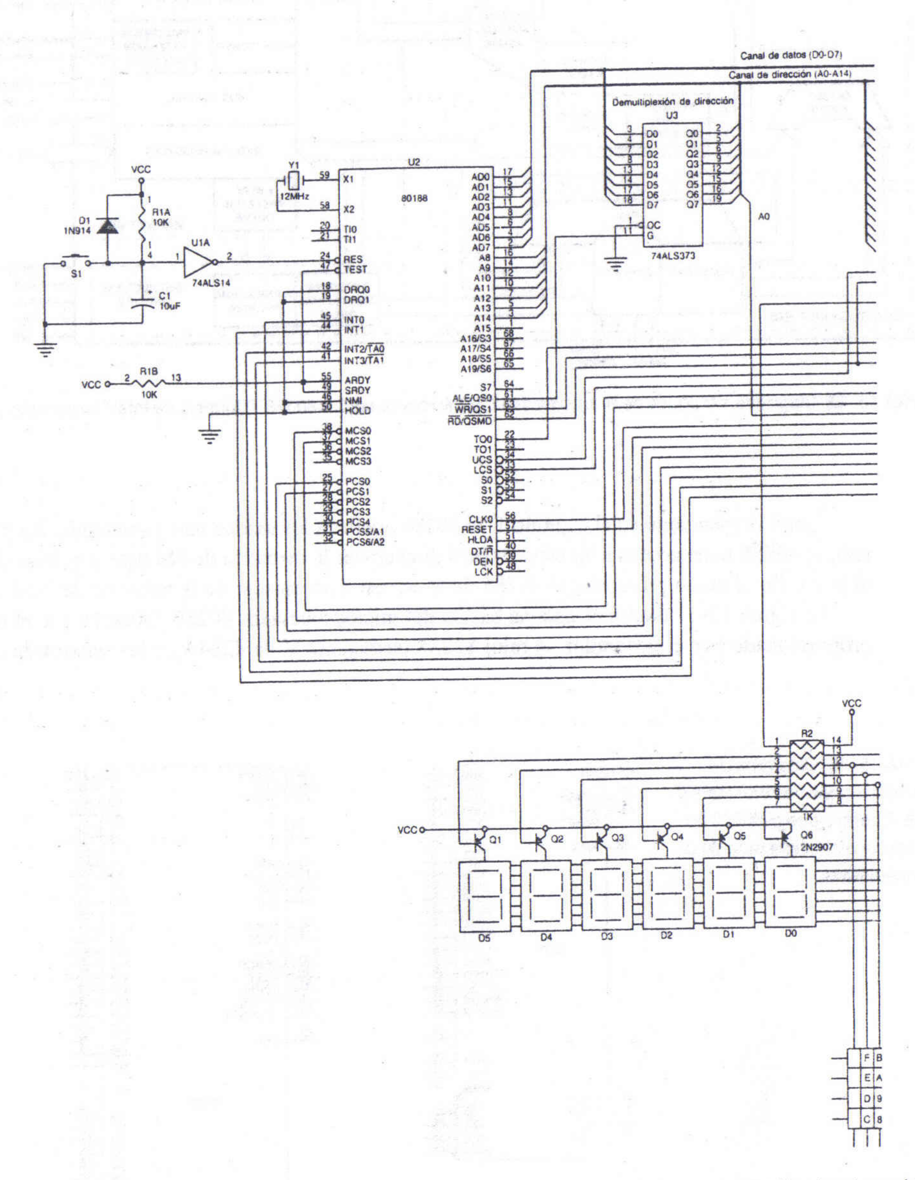
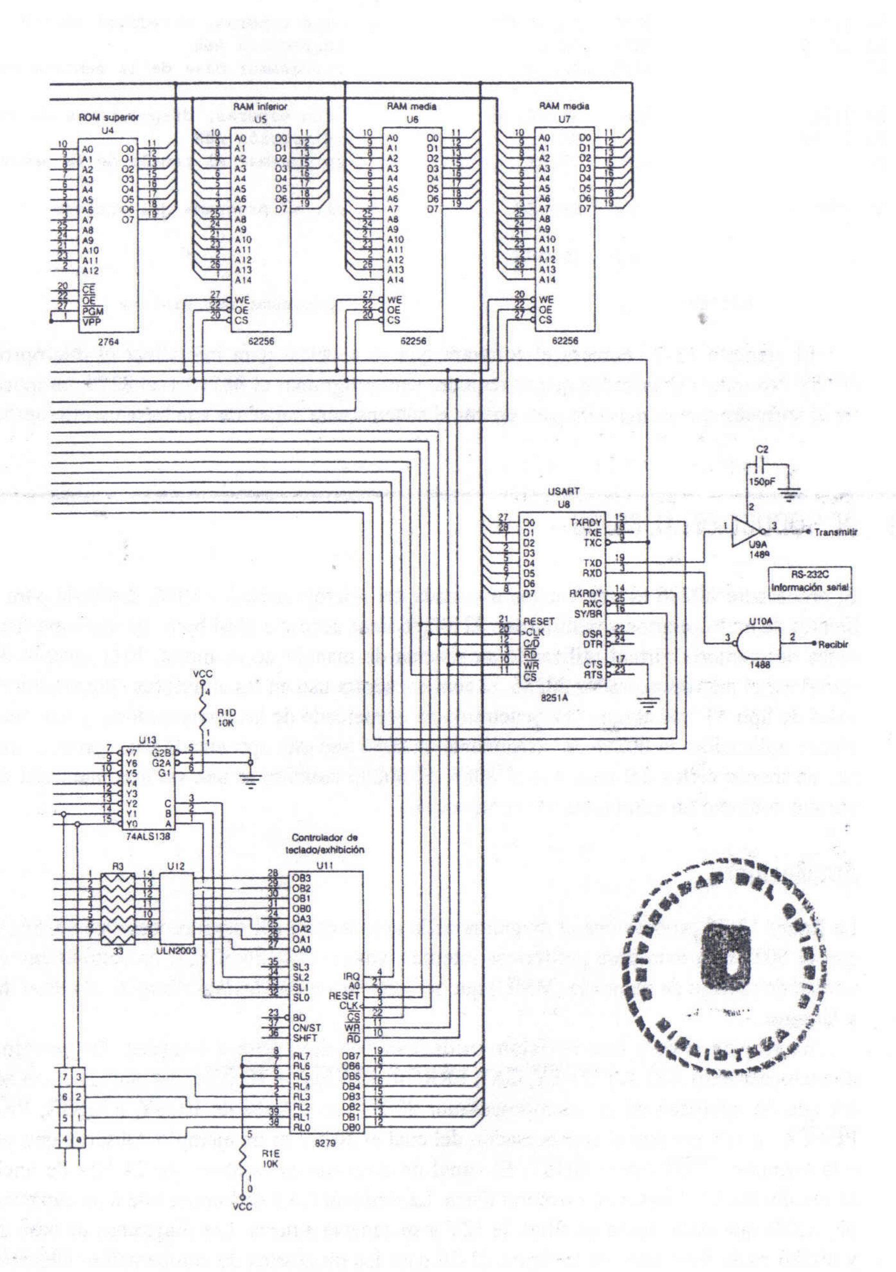


FIGURA 13–24 Un sistema basado en el 80188 que contiene un interface para teclado, una exhibición visual numérica de 6-dígitos, y una interface serial.


ROM superior
U4
2764
RAM inferior
U5
62256
RAM media
U6
62256
RAM media
U7
62256
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
O0
O1
O2
O3
O4
O5
O6
O7
D0
D1
D2
D3
D4
D5
D6
D7
CE
OE
PGM
VPP
WE
OE
CS
C2
150pF
USART
U8
8251A
RESET
CLK
C/D
RD
WR
CS
TXRDY
TXE
TXC
TXD
RXD
RXRDY
RXC
SY/BR
DSR
DTR
CTS
RTS
U9A
1489
Transmitir
RS-232C
Información serial
U10A
1488
Recibir
VCC
R1D
10K
U13
74ALS138
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
G2B
G2A
G1
C
B
A
Controlador de
teclado/exhibición
U11
8279
R3
33
U12
ULN2003
OB3
OB2
OB1
OB0
OA3
OA2
OA1
OA0
SL3
SL2
SL1
SL0
BD
CN/ST
SHFT
RL7
RL6
RL5
RL4
RL3
RL2
RL1
RL0
IRQ
A0
RESET
CLK
CS
WR
RD
DB7
DB6
DB5
DB4
DB3
DB2
DB1
DB0
R1E
10K
VCC

```
FF8E B8 103E          MOV   AX,103EH       ;2 esperas, dirección 1000H
FF91 83 C2 02         ADD   DX,2           ;dirección A4H
FF94 EF               OUT   DX,AX          ;programa E/S

FF95 B8 81FC          MOV   AX,81FCH       ;sin esperas, dirección 80000H
FF98 83 C2 02         ADD   DX,2           ;dirección A6H
FF9B EF               OUT   DX,AX          ;programar base de la memoria media

FF9C B8 903C          MOV   AX,903CH       ;sin esperas, dispositivos de 32K
FF9F 83 C2 02         ADD   DX,2           ;dirección A8H
FFA2 EF               OUT   DX,AX          ;programar el tamaño de la memoria media

FFA3 E9 F000 R        JMP   SYSTEM         ;ir al programa del sistema

F000                  ORG   0F000H

F000          SISTEMA:                     ;programa del sistema
```

El ejemplo 13-3 enumera el software que se necesita para inicializar el microprocesador 80188. No indica el software que se requiere para programar el 8851A o el 8279, tampoco muestra el software que se necesita para operar el sistema para capacitar con base en microprocesador.

13–4 INTRODUCCION AL 80286

El procesador 80286 es una versión avanzada del microprocesador 8086, diseñada para los ambientes de multiusuarios y multitareas. El 80286 tiene acceso a 16M bytes de memoria física y 1G bytes de memoria virtual utilizando su sistema de manejo de memoria. Esta sección del texto introduce el microprocesador 80286, el cual encuentra uso en las anteriores computadoras personales de tipo AT que alguna vez penetraron en el mercado de las computadoras y aún encuentran alguna aplicación. El 80286 es básicamente un 8086 que está optimizado para ejecutar instrucciones en menos ciclos del reloj que el 8086. El 80286 también es una versión mejorada del 8086 porque contiene un administrador de memoria.

Arquitectura

La figura 13-25 proporciona el diagrama de la arquitectura del microprocesador 80286. Observe que el 80286 no incorpora periféricos internos, como el 80186/80188; en cambio contiene una unidad de manejo de memoria (MMU) que se llama la *unidad de direccionamiento* en el diagrama a bloques.

Conforme revela una revisión cuidadosa del diagrama a bloques, las terminales de direccionamiento A23-A0, $\overline{BUSY}$, CAP, $\overline{ERROR}$, $\overline{PEREQ}$, y $\overline{PEACK}$ son nuevas o son adicionales que no aparecen en el microprocesador 8086. Las señales de $\overline{BUSY}$, $\overline{ERROR}$, $\overline{PEREQ}$, y $\overline{PEACK}$ se utilizan con el coprocesador, del cual el 80287 es un ejemplo. Observe que se refiere a la terminal $\overline{TEST}$ como $\overline{BUSY}$. El canal de direcciones es ahora de 24 bits de ancho para acomodar los 16M bytes de memoria física. La terminal CAP está conectada a un capacitor 0.047 μF, ±20% que actúa como un filtro de 12V y se conecta a tierra. Los diagramas de base del 8086 y 80286 están ilustrados en la figura 13-26 para los propósitos de comparación. Observe que el 80286 no contiene un canal multiplexado de dirección/datos.

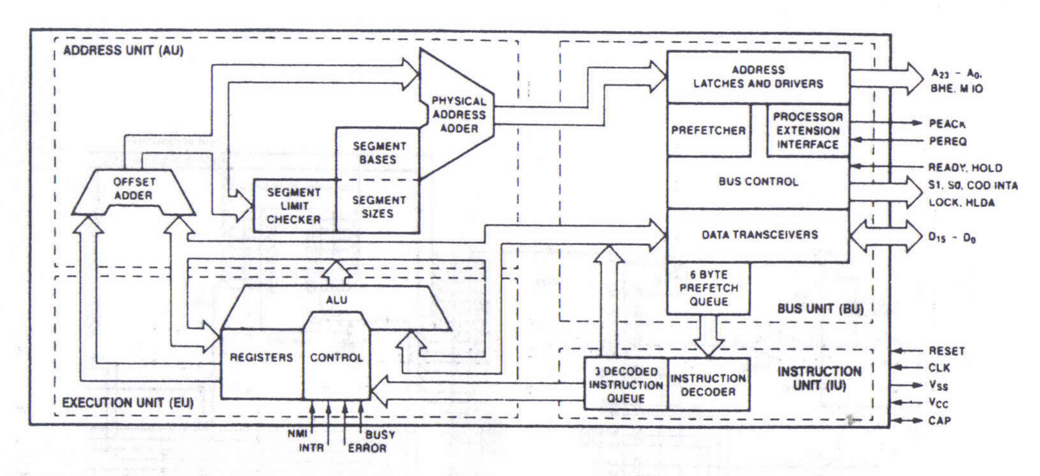
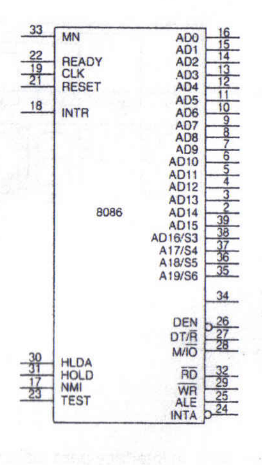


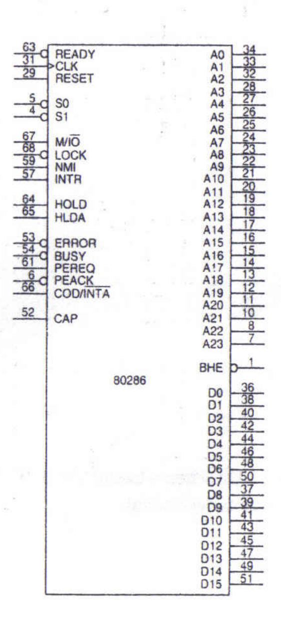
FIGURA 13-25 Diagrama a bloques de la arquitectura del microprocesador 80286. (Cortesía de Intel Corporation.)

Como se mencionó en el capítulo 1, el 80286 opera en los modos real y protegido. En el modo real, el 80286 tiene acceso a un espacio para direccionar la memoria de 1M byte y es casi idéntico al 8086. En el modo protegido, el 80286 tiene acceso a un espacio de la memoria de 16M byte.

La figura 13-27 ilustra el sistema básico del microprocesador 80286. Observe que el reloj es proporcionado por el generador de reloj 82284 (semejante al de 8284A) y las señales de control

FIGURA 13-26 Diagrama de base del microprocesador 8086 y 80286. Observe que el 80286 no tiene un canal multiplexado de direcciones/datos.





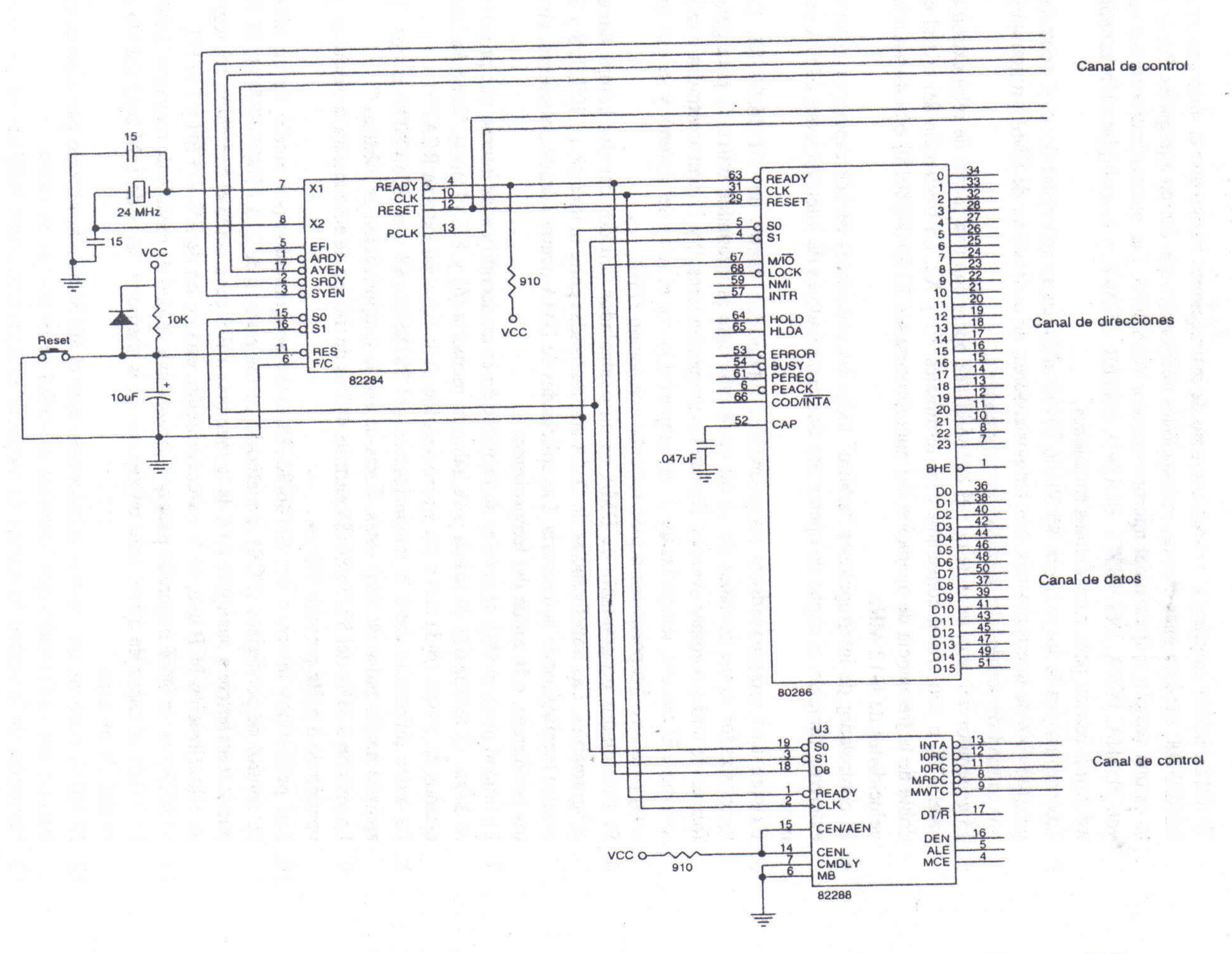


FIGURA 13–27 La interconexión del microprocesador 80286, con el generador de reloj 82284, y el controlador del canal del sistema 8288.

del sistema son proporcionadas por el controlador del canal del sistema 82288 (semejante al 8288). Observe también la ausencia de los registros utilizados para demultiplexar el canal de direcciones/datos del 8086.

Instrucciones adicionales

El 80286 tiene aún más instrucciones que sus antecesores. Estas instrucciones adicionales controlan el sistema de la memoria virtual por medio del administrador de memoria del 80286. La tabla 13-7 muestra las instrucciones adicionales del 80286 con un comentario sobre el propósito de cada instrucción. Estas son las únicas instrucciones nuevas agregadas al 80286. Observe que el 80286 también contiene las nuevas instrucciones agregadas al 80186/80188, tal como INS, OUTS, BOUND, ENTER, LEAVE, PUSHA, POPA, y la multiplicación inmediata así como cuenta inmediata para corrimientos y rotaciones.

A continuación están las descripciones de las instrucciones que no fueron explicadas dentro de la sección de manejo de memoria. Las instrucciones descritas aquí son especiales en su naturaleza y sólo son utilizadas para las condiciones indicadas.

CLTS. La instrucción de desactivación de la bandera de conmutación de tarea (CLTS) borra (desactiva) el bit de la bandera TS (conmutación de tareas) con un 0 lógico. Si la bandera TS es un 1 lógico, y el coprocesador numérico 80287 se usa para la tarea, ocurre una interrupción (vector 7). Esto permite que la función del coprocesador sea emulada por programa. La instrucción CLTS será usada en un sistema y se considerará una instrucción privilegiada porque sólo se puede ejecutar en el nivel 0 del modo protegido al nivel 0. No hay ninguna instrucción para activar (1) la bandera TS. Esto se logra escribiendo un 1 lógico en la posición del bit 3 (TS) de la palabra de estado de la máquina (MWS) utilizando la instrucción LMSW.

TABLA 13-7 Instrucciones adicionales del 80286

Instrucción	*Comentario*
CLTS	Desactiva la bandera de conmutación de tarea
LDGT	Carga al registro para la tabla del descriptor global
SGDT	Almacena el registro de la tabla de descriptores globales
LIDT	Carga al registro de la tabla del descriptor del conmutador
SIDT	Almacena al registro de la tabla del descriptor del conmutador
LLDT	Carga al registro de la tabla del descriptor local
SLDT	Almacena al registro de la tabla del descriptor local
LMSW	Carga la palabra del estado de la máquina
SMSW	Almacena la palabra del estado de la máquina
LAR	Carga los derechos de acceso
LSL	Carga el límite del segmento
SAR	Almacena los derechos de acceso
ARPL	Ajusta el nivel del privilegio solicitado
VERR	Verifica acceso a lectura
VERW	Verifica accesó a escritura

LAR. Carga derechos de acceso (LAR) lee el segmento descriptor y coloca una copia del byte de los derechos de acceso en un registro de 16-bits. Un ejemplo es la instrucción LAR AX,BX que carga a AX con el byte de derechos de acceso del descriptor seleccionado por el valor del selector dado en BX. Esta instrucción se utiliza para obtener los derechos de acceso de manera que pueda ser revisada antes de que un programa use el segmento de la memoria que describe el descriptor.

LSL. Instrucción límite de carga del segmento (LSL) carga un registro especificado a usuario con el límite del segmento. Por ejemplo, la instrucción LSL AX,BX carga en AX el límite del segmento indicado por el descriptor seleccionado por el selector dado BX. Esta instrucción se usa para probar el límite de un segmento.

ARPL. La instrucción de ajuste del nivel del privilegio solicitado (ARPL) se usa para probar un selector para que el nivel de privilegio del selector solicitado no sea violado. Un ejemplo es ARPL AX,CX en donde AX contiene el nivel de privilegio solicitado y CX contiene el valor del selector que deberá usarse para tener acceso a un descriptor. Si el nivel de privilegio solicitado es de una prioridad más baja que el que está bajo prueba, la bandera cero se activa. Esto puede requerir que un programa ajuste el nivel de privilegio solicitado o indique una violación de privilegios.

VERR. Instrucción para verificar para acceso a lectura (VERR) verifica que un segmento se puede leer. Recuerde que se dijo en el capítulo 1 que un segmento de código puede estar protegido contra lectura. Si el segmento de código se puede leer, el bit de la bandera cero se activa. La instrucción VERR AX prueba el descriptor seleccionado por el registro AX.

VERW. Instrucción para verificar el acceso a escritura (VERW) se usa para verificar que se puede escribir en un segmento. Recuerde que en el capítulo 1 un segmento de datos puede estar protegido contra escritura. Si se puede escribir en el segmento de datos, el bit de la bandera cero se activa.

La máquina de memoria virtual

Una *máquina de memoria virtual* es una máquina que distribuye un espacio de la memoria más grande (1G byte para el 80286) en un espacio físico de memoria mucho más pequeño (16M bytes para el 80286). Esto permite que un sistema muy grande se ejecuta en sistemas con memoria física más pequeña. Esto se logra intercambiando los datos y los programas entre el disco duro del sistema de memoria y la memoria física. Direccionar un sistema de memoria de 1G byte se logra con los descriptores del microprocesador 80286. Cada descriptor del 80286 indica un segmento de memoria de 64K bytes y el 80286 permite hasta 16k descriptores. Esto (64K × 16K) permite que hasta un máximo de 1G byte de memoria sea descrito por el sistema.

Cómo se mencionó en el capítulo 1, los descriptores indican el segmento de la memoria en el modo protegido. El 80286 tiene descriptores que definen código, datos, y los segmento de pila, interrupciones, procedimientos, y tareas. Los accesos a los descriptores se realizan cargando un registro de segmentos con un selector en el modo protegido. El selector accesa al descriptor que describe un área de la memoria. Detalles adicionales sobre descriptores y su aplicación son definidos en el capítulo 1 y en el capítulo 14. Refiérase a estas secciones del texto para una descripción mucho más detallada del sistema de manejo de memoria del modo protegido.

13–5 RESUMEN

1. El 80186/80188 contiene el mismo conjunto de instrucciones básico que el microprocesador 8086/8088, excepto unas cuantas instrucciones adicionales que fueron agregadas. Por lo tanto es una versión mejorada del microprocesador 8086/8088. Las nuevas instrucciones incluyen PUSHA, POPA, INS, OUTS, BOUND, ENTER, LEAVE, y la multiplicación inmediata, así como conteo para corrimientos/rotaciones.
2. Las mejorías en la arquitectura del 80186/80188 incluyen un generador de reloj, controlador programable de interrupciones, tres temporizadores, un controlador de DMA programable, y una unidad de señales de habilitación programable.
3. El generador del reloj permite que el 80186/80188 opere con una fuente de reloj extrema de nivel TTL o con cristal conectado a las terminales X1 y X2. La frecuencia del cristal es el doble de la frecuencia de operación del microprocesador. El 80186/80188 está disponible en velocidades de 6-12 MHz.
4. El controlador de interrupciones "arbitra" todas las solicitudes de interrupciones internas y externas. También es capaz de operar con dos controladores de interrupciones 8259A externos.
5. Existen tres temporizadores programables integrados dentro del 80186/80188. Cada temporizador es un contador de 16-bit completamente programable utilizado para generar formas de ondas o contar eventos. Dos de los temporizadores, 0 y 1, tienen entradas y salidas externas. El tercero, temporizador 2, es temporizado por el reloj del sistema y se usa como reloj para otro temporizador o para solicitar una acción DMA.
6. El controlador programable de DMA es un controlador con dos canales completamente programables. Las transferencias de DMA son realizadas entre la memoria y E/S, E/S y E/S, o entre localidades de la memoria. Las solicitudes de DMA ocurren desde programas, circuitos periféricos, o la salida del temporizador 2.
7. La unidad programable de señales de habilitación es un decodificador interno que proporciona hasta 13 terminales de salida para habilitar memoria (6) y E/S (7 pins). También inserta estados de espera (0-3) con o sin sincronización de la terminal externa READY.
8. La única diferencia entre la temporización del 80186/80188 y el 8086/8088 es que ALE aparece medio pulso de reloj antes. Fuera de eso la temporización es idéntica.
9. La versión 6 MHz del 80186/80188 permite 417 ns de tiempo de acceso para la memoria, y la versión de 8 MHz permite 309 ns.
10. Los periféricos internos del 80186/80188 están programados por medio de un bloque de control de periféricos (PCB) inicializado en los puertos de E/S FF00H-FFFFH. El PCB puede transferirse a cualquier área de la memoria o E/S cambiando el contenido del registro de relocalización PCB que, en la reinicialización, está en las de E/S FFFEH y FFFFH.
11. El 80286 es un 8086 mejorado para que incluya una unidad de manejo de memoria (MMU). El 80286 es capaz de direccionar un espacio de la memoria física de 16M byte debido a la unidad de manejo.
12. El 80286 contiene las mismas instrucciones que el 80186/80188, excepto por unas cuantas instrucciones adicionales que controlan la unidad de manejo de memoria.
13. Por medio de la unidad de manejo de memoria, el microprocesador 80286 tiene acceso a un espacio virtual de la memoria de 1G byte, como fue especificado por los 16K archivados en las dos tablas.

13–6 CUESTIONARIO Y PROBLEMAS

1. Mencione las diferencias entre el microprocesador 8086/8088 y el 80186/80188.
2. ¿Cuáles mejorias fueron agregadas a la arquitectura del 80186/80188 y no están presentes en el 8086/8088?
3. ¿El 80186/80188 está encapsulado en qué tipos de circuito integrado?
4. Si el cristal de 20 MHz está conectado a X1 y X2, ¿cuál señal de frecuencia se encuentra en CLKOUT?
5. ¿Cómo se hace disponible el estado de la cola de espera del coprocesador numérico 80187 en las terminales ALE y $\overline{WR}$?
6. La capacidad de corriente de salida de cualquier terminal del 80186/80188 excepto para CLKOUT y $\overline{S2}/\overline{S0}$ es de ______ para un 0 lógico.
7. ¿Cuántos periodos de reloj hay en el ciclo del canal de 80186/80188?
8. ¿Cual es la diferencia principal entre la temporización del 8086/8088 y el 80186/80188?
9. ¿Cuál es la importancia del tiempo de acceso a la memoria?
10. ¿Cuánto tiempo de acceso a la memoria es permitido por el 80186/80188 si es operado con un reloj de 6 MHz?
11. ¿En dónde está ubicado el bloque de control de periféricos después de que el 80186/80188 se reinicializa?
12. Escriba un programa para relocalizar al bloque de control periférico a las localidades 10000H-100FFH de la memoria.
13. ¿Cual vector de interrupción es utilizado por la terminal INTO en el microprocesador 80186/80188?
14. ¿Cuantos vectores de interrupción están disponibles para el controlador de interrupción del microprocesador 80186/80188?
15. ¿Cuáles dos modos de operación están disponibles en el controlador de interrupciones?
16. ¿Cuál es el propósito del registro de control de interrupciones?
17. Siempre que la fuente de una interrupción está enmascarada, el bit de enmascaramiento en el registro de enmascaramiento del controlador de interrupciones es un _____lógico.
18. ¿Cual es la diferencia entre los registros de encuesta de interrupciones y el de estado de encuesta?
19. ¿Cuál es el propósito del registro para fin de interrupción (EOI)?
20. ¿Cuántos temporizadores de 16 bits tiene el 80186/80188?
21. ¿Cuáles temporizadores tienen terminales de entrada y salida?
22. ¿Cuál temporizador se conecta al reloj del sistema?
23. Si dos registros de cuenta máxima son utilizados en un temporizador, explique la operación del temporizador.
24. ¿Cuál es el propósito del bit del registro de control del temporizador INH?
25. ¿Cuál es el propósito del bit de registro de control del temporizador P?
26. ¿El bit del registro de control del temporizador ALT selecciona qué tipo de operación para los temporizadores 0 y 1?
27. Explique cómo se utilizan las salidas del temporizador.
28. Desarrolle un programa que haga que el temporizador 1 genere una señal continua que sea de un 1 lógico para 123 cuentas y un 1 lógico para 23 cuentas.

29. Desarrolle un programa que ocasione que el temporizador 0 genere un solo pulso después de que 345 pulsos de reloj hayan ocurrido en su terminal de entrada.
30. ¿Cuántos canales de DMA son controlados por el controlador de DMA?
31. El registro fuente del controlador de DMA y el registro destino son cada uno de ________ bits de ancho.
32. ¿Cómo se arranca el canal de DMA por programa?
33. La unidad de señales de habilitación tiene ______ terminales para habilitación de dispositivos de memoria.
34. La unidad de señales de habilitación tiene _______ terminales para periféricos.
35. La última localidad del bloque superior de la memoria que habilita la terminal $\overline{UCS}$ es la localidad ________.
36. Las terminales de habilitación de la memoria media están programados para una dirección __________ y tamaño de bloque.
37. El área de memoria inferior seleccionada por $\overline{LCS}$ comienza en la dirección __________.
38. El generador interno del estado de espera es capaz de insertar estados de espera entre ________ y __________.
39. Programe al registro en el desplazamiento A8H del bloque de memoria media sea de 128K bytes.
40. ¿Cuál es el propósito del bit EX en el registro en el desplazamiento A8H?
41. El microprocesador 80286 direcciona _______ bytes de memoria física.
42. Cuando el administrador de la memoria se está usando, el 80286 direcciona _______ bytes de memoria virtual.
43. El conjunto de instrucciones del 80286 es idéntico al del ________, excepto por las instrucciones de manejo de memoria.
44. ¿Cuál es el propósito de la instrucción VERR?
45. ¿Cuál es el propósito de la instrucción LSL?

CAPITULO 14

Los microprocesadores 80386 y 80486

INTRODUCCION

El microprocesador 80386 es una versión completa de 32 bits del microprocesador 80286. Además de tener un tamaño de palabra más grande (doble palabra), contiene muchas mejoras y características adicionales. El microprocesador 80386 cuenta con multitareas, administración de memoria, memoria virtual con o sin paginación, protección de programación, y un sistema de memoria grande. Toda la programación escrita para las versiones anteriores 8086/8088, 80186/80188 y el 80286 son compatibles, es decir, son "escalables" para el microprocesador 80386. La cantidad de memoria que puede direccionar el 80386 se incrementa a 4G bytes en comparación con los 1M bytes encontrados en el 8086/8088/80186/80188 y los 16M bytes encontrados en el 80286. El 80386 se puede cambiar entre el modo protegido y el modo real sin reinicializar el microprocesador. En el microprocesador 80286 el cambiar del modo protegido al modo real es un problema.

El microprocesador 80486 es una versión mejorada del microprocesador 80386 que ejecuta muchas de sus instrucciones sólo un periodo de reloj, en lugar de los dos que requiere el 80386. El microprocesador 80486 también contiene una memoria caché de 8K bytes y una versión mejorada del coprocesador numérico 80387. Cuando el 80486 se opera con la misma frecuencia de reloj que el 80386, funciona con una mejoría en la velocidad de aproximadamente el 50 por ciento.

OBJETIVOS DEL CAPITULO

Una vez que concluya este capítulo, el lector podrá:

1. Realizar una comparación del microprocesador 80386 con los otros anteriores de la familia.
2. Describir la organización y la interface del sistema de memoria de 32 bits del 80386.
3. Describir la operación de la unidad de administración de la memoria y de la unidad de paginación del 80386.
4. Cambiar del modo protegido al modo real.
5. Definir la operación de las instrucciones adicionales y modos de direccionamiento del 80386.

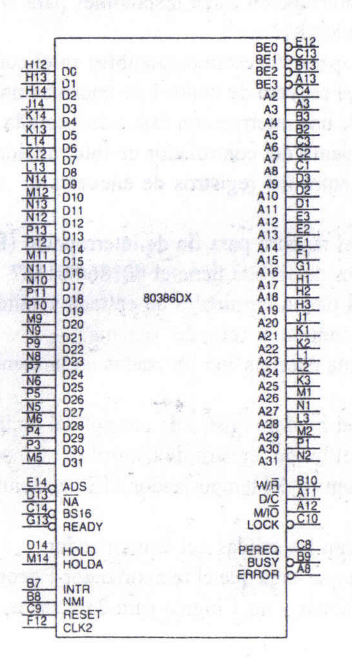
6. Explicar la operación de un sistema con memoria caché y un sistema con memoria entrelazada.
7. Detallar la estructura de las interrupciones y la estructura de acceso directo a memoria del 80386.
8. Hacer una comparación del 80486 con los microprocesadores 80386 y 80286.
9. Detallar la operación de las nuevas instrucciones del 80486.
10. Explicar la operación de la memoria caché del 80486.

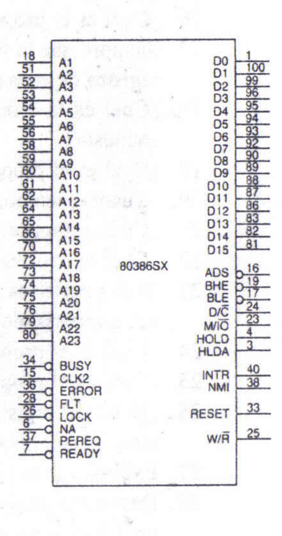
14-1 INTRODUCCION AL MICROPROCESADOR 80386

Antes que se pueda utilizar este microprocesador en un sistema, se debe comprender la función de cada terminal. Esta sección del capítulo detalla la operación de cada una de éstas junto con la estructura de los registros internos y del sistema externo de memoria y de la estructura de entrada/salida del microprocesador 80386.

La figura 14-1 muestra el diagrama de base del microprocesador 80386DX que está encapsulado en un PGA (rejilla de terminales) de 132 terminales y del microprocesador 80386SX. Estos microprocesadores también están disponibles en el encapsulado para montaje de superficie que es más pequeño. Por lo regular se dispone de dos versiones del 80386: el 80386DX, que se muestra y describe en este capítulo, es la versión completa y el 80386SX que es una versión con el canal reducido. También está disponible una nueva versión del 80386: el 80386SL, que incorpora la mayor parte del sistema del canal AT. El 80386DX direcciona 4G bytes de memoria a través de su

FIGURA 14-1 Diagrama de base del microprocesador 80386DX y del 80386SX.





canal de datos de 32 bits y su canal de direcciones de 32 bits. El 80386SX, más parecido al 80286, direcciona 16 M bytes de memoria con su canal de direcciones de 24 bits por medio de su canal de datos de 16 bits. El 80386SX se desarrolló después del 80386DX para aplicaciones que no requerían la versión completa del canal de 32 bits. El 80386SX se encuentra en muchas computadoras personales que utilizan el mismo diseño básico de tarjeta madre del 80286. En la actualidad, la mayor parte de las aplicaciones necesitan menos de 16M bytes de memoria, por lo que el 80386SX es una versión bastante popular y menos costosa del microprocesador 80386.

Los miembros anteriores de la familia necesitan una alimentación de +5.0 V al igual que el microprocesador 80386. (Observe que algunos de los microprocesadores más recientes para las computadoras tipo notebook utilizan una alimentación de +3.0 V.) El consumo de corriente varía entre 550 mA para la versión de 25 MHz del 80386, 500 mA para la versión de 20 MHz, y 450 mA para la versión de 16 MHz. También está disponible una versión de 33 MHz que consume 600 mA de corriente. Observe que durante algunos modos de operación normal, se consume de la fuente de alimentación a más de 1.0 A. Esto significa que la fuente de alimentación y la red de distribución de energía tienen que ser capaces de suministrar estas cargas. Este equipo contiene múltiples conexiones de Vcc y Vss que deben estar todas conectadas a +5.0 V y a tierra para una operación adecuada. Algunas de las terminales están etiquetadas como N/C (no conexión) y no se deben conectar.

Cada terminal de salida del 80386 es capaz de disipar (0 lógico) 4.0 mA (conexiones de direcciones y datos) o 5.0 mA (otras conexiones). Esto representa un incremento en el manejo de corriente en comparación a los 2.0 mA disponibles en las terminales de salida de miembros anteriores de la familia. Cada terminal de entrada representa una pequeña carga que requiere de sólo ±10 μA de corriente. En casi todos los sistemas, excepto los más pequeños, estos niveles de corriente requieren de acopladores del canal.

La función de cada grupo de terminales del 80386 es como sigue:

1. A31 –A2 — Conexiones del canal de direcciones: se utilizan para direccionar cualquiera de las localidades de la memoria de 1G × 32 que se encuentran en el sistema de memoria del 80386. Observe que A0 y A1 están codificados en la habilitación del canal (BE3–BE0) descrito en otra parte del texto. (Observe que el 80386SX contiene conexiones de direccionamiento A23–A1.)
2. D31–D0 — Conexiones del canal de datos: se utilizan para transferir datos entre el microprocesador y los sistemas de memoria y de entrada/salida. (Observe que el 80386SX contiene conexiones D15–D0 del canal de datos.)
3. $\overline{BE3}$ –$\overline{BE0}$ — Señales de habilitación de banco: se utilizan para direccionar un byte, una palabra, o doble palabra de datos. Estas señales las genera internamente el microprocesador de los bits de dirección A1 y A0. (Observe que 80386SX contiene $\overline{BHE}$ y $\overline{BLE}$ para la selección de banco.)
4. M/$\overline{IO}$ — Memoria E/S: selecciona un dispositivo de memoria cuando es un 1 lógico o un dispositivo de entrada/salida cuando es un 0 lógico. Durante la operación de entrada/salida el canal de direcciones contiene una dirección de entrada/salida de 16 bits.
5. W/$\overline{R}$ — Escritura/Lectura: indica que el ciclo actual del canal de escritura cuando es un 1 lógico o de lectura cuando es un 0 lógico.
6. $\overline{ADS}$ — La función estroboscópica de datos dirección: se activa cuando el 80386 proporciona una dirección válida de memoria o de entrada/salida. Esta señal se combina con la señal W/$\overline{R}$ para generar las señales separadas de lectura y escritura que se encuentran presentes en los primeros sistemas 8086/8088.

7. RESET — Reinicializar: reinicializa al 80386 haciendo que empiece a ejecutar desde la localidad de memoria FFFFFFF0H. El 80386 se reinicializa en el modo real y los 12 bits más a la izquierda de la dirección permanecen como unos lógicos (FFFH) hasta que se ejecute un brinco o llamada lejana cuando se hacen ceros lógicos (000H).
8. CLK2 — 2 Veces de reloj: la maneja una señal de reloj que es el doble de la frecuencia de operación del 80386. Por ejemplo, para operar el 80386 a 16 MHz, aplicamos un reloj de 32 MHz a esta terminal.
9. $\overline{\text{READY}}$ — Listo: controla el número de estados de espera insertados en el ciclo de acceso a la memoria.
10. $\overline{\text{LOCK}}$ — Prefijo: es un 0 lógico cuando una instrucción tiene el prefijo LOCK:. Esto se utiliza con más frecuencia durante los accesos de DMA.
11. D/$\overline{\text{C}}$ — Datos/Control: indica que el canal de datos contiene datos para o de la memoria o entrada/salida cuando es un 1 lógico. Si D/$\overline{\text{C}}$ es un 0 lógico, el microprocesador se detiene o se ejecuta un reconocimiento de interrupción.
12. $\overline{\text{BS16}}$ — Tamaño del canal 16 bits: selecciona un canal de datos de 32 bits ($\overline{\text{BS16}}$ = 1) o un canal de datos de 16 bits ($\overline{\text{BS16}}$ = 0). En casi todos los casos, si un 80386 se opera con un canal de datos de 16 bits, se utiliza el 80386SX que cuenta con un canal de datos de 16 bits.
13. $\overline{\text{NA}}$ — Dirección siguiente: hace que el 80386 dé salida a la dirección de la siguiente instrucción o dato en el ciclo actual del canal. Se utiliza con frecuencia para trabajar en paralelo la dirección.
14. HOLD: solicita una acción DMA como lo hizo con el 8086/8088.
15. HLDA — Reconocer HOLD: indica que el 80386 se encuentra actualmente en posición de ceder su canal.
16. $\overline{\text{PEREQ}}$ — Solicita coprocesador: le pide al 80386 que deje el control, y es una conexión directa al coprocesador aritmético 80387.
17. $\overline{\text{BUSY}}$ — Ocupado: una entrada utilizada por la instrucción WAIT o FWAIT que espera que el coprocesador se desocupe. También es una conexión directa al 80387 del 80386. Funciona como la terminal $\overline{\text{TEST}}$ del 8086/8088.
18. $\overline{\text{ERROR}}$ — Error: le indica al microprocesador que el coprocesador detectó un error.
19. INTR — Solicita una interrupción: lo utilizan los circuitos externos para solicitar una interrupción.
20. NMI — Interrupción no enmascarable: solicita una interrupción no enmascarable como lo hizo con el microprocesador 8086.

Sistema de la memoria

El sistema físico de la memoria del 80386DX es de 4G bytes y se puede direccionar como tal, o si se utiliza el direccionamiento virtual, se mapean hasta 64T bytes en los 4G bytes de espacio físico de la unidad de manejo de la memoria. La figura 14-2 muestra la organización del sistema físico de memoria del 80386DX.

La memoria se divide en cuatro bancos de memoria de 8 bits de ancho, los cuales contienen cada uno hasta 1G bytes de memoria. Una organización de memoria como ésta de 32 bits de ancho, permite que se accesen directamente bytes, palabras o dobles palabras. El 80386DX transfiere un número de hasta 32 bits de ancho en un solo ciclo de memoria, mientras que el 8088 anterior requiere de 4 ciclos para realizar la misma transferencia y el 80286 requiere de 2 ciclos. En la actualidad, este ancho de datos es importante, sobre todo para los números con punto

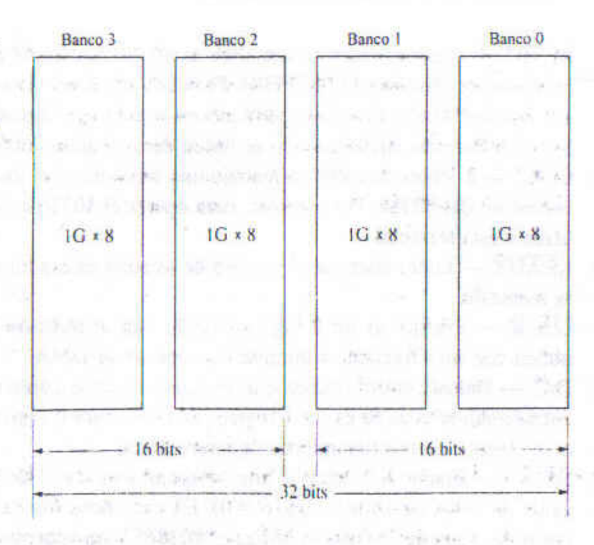


FIGURA 14–2 El sistema de memoria para el microprocesador 80386. Observe que la memoria está organizada en 4 bancos, los cuales contienen cada uno 1G bytes. La memoria se accesa como datos de 8, 16 o 32 bits.

decimal flotante de precisión sencilla, que son de 32 bits de ancho y que se utilizan con programas para gráficas. Por lo regular, los lenguajes de alto nivel utilizan números con punto decimal flotante para el almacenamiento de datos, así que las localidades de memoria de 32 bits agilizan la ejecución de programas en lenguaje de alto nivel, si están escritos para aprovechar esta memoria más ancha.

Cada byte de memoria está numerado en hexadecimal como en las versiones anteriores de la familia. La diferencia es que el 80386DX utiliza una dirección de memoria de 32 bits de ancho con los bytes de memoria numerados desde la localidad 00000000H hasta la FFFFFFFFH.

Los dos bancos de memoria en el sistema del 8086 se accesan por medio de A0 y $\overline{\text{BHE}}$. En el 80386DX, los bancos de memoria se accesan por medio de cuatro señales de habilitación de banco $\overline{\text{BE0}}$-$\overline{\text{BE3}}$. Este arreglo permite que se accese un solo byte cuando el microprocesador activa sólo una señal de habilitación de banco. También permite que se direccione una palabra cuando dos señales de habilitación de banco se activan. En la mayor parte de los casos, se direcciona una palabra en el banco 0 y 1 o en el banco 2 y 3. La localidad 00000000H de la memoria se encuentra en el banco 0, la localidad 00000001H de la memoria se encuentra en el banco 1, la 00000002H está en el banco 2, y la localidad 00000003H está en el banco 3. El 80386DX no contiene las terminales de direccionamiento A0 y A1 de dirección porque han sido codificadas como señales de habilitación de banco.

Sistema con acopladores. La figura 14-3 muestra el 80386DX conectado a acopladores que incrementan la capacidad de corriente de salida de sus conexiones de direcciones, datos y control. Este microprocesador se opera a 25 MHz utilizando una señal de entrada de reloj de 50 MHz que genera un módulo oscilador integrado. Los módulos osciladores casi siempre se utilizan para proporcionar el reloj en equipos modernos que utilizan microprocesadores. La señal HLDA se conecta a la entrada de control de habilitación de los acopladores en un sistema que utiliza acceso directo a memoria. De lo contrario, las terminales de habilitación de los acopladores se conectan a tierra en un sistema sin DMA.

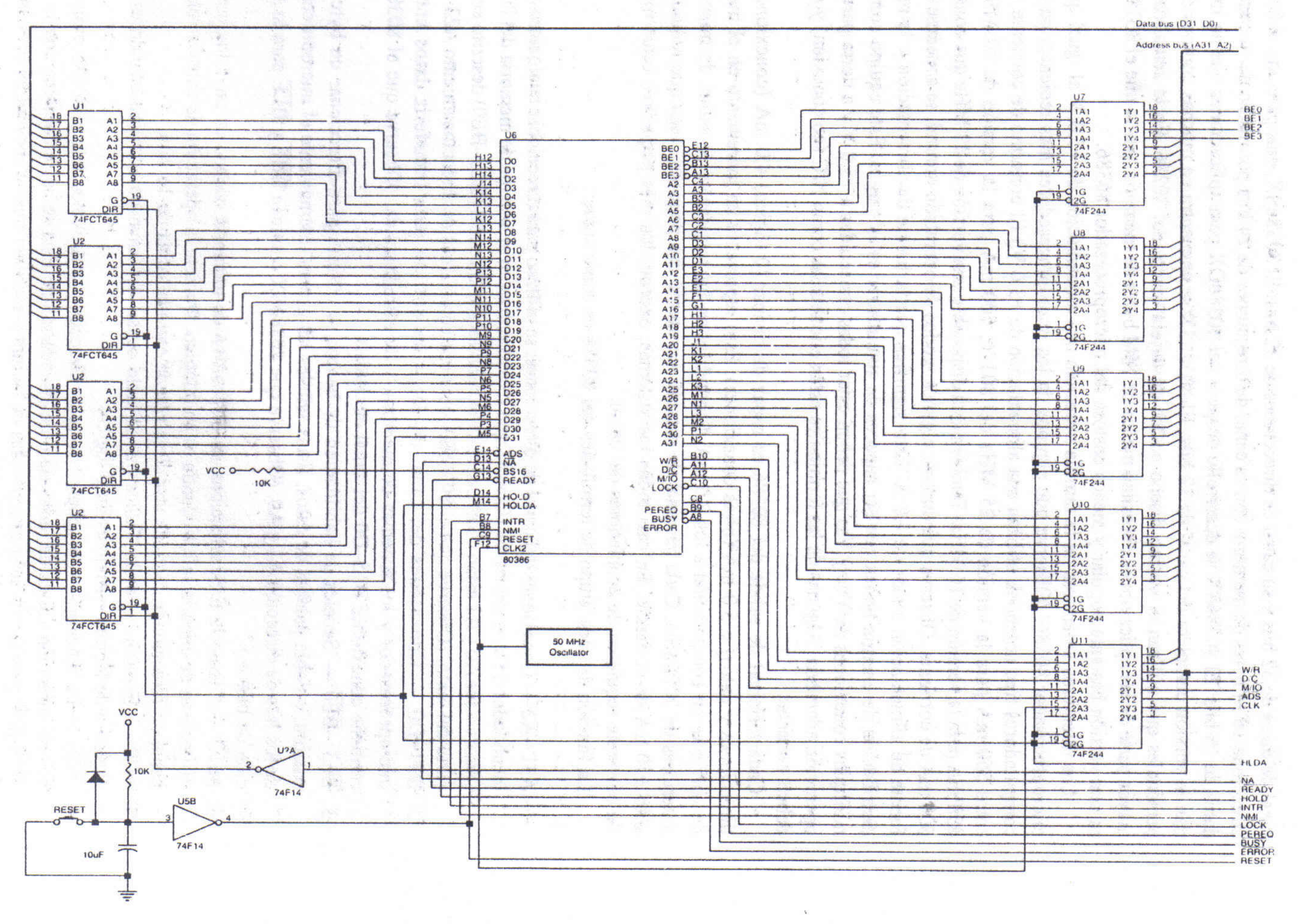


FIGURA 14–3 Un 80386DX de 25 MHz completamente acoplado.

Paralelismo entrelazado y cachés. La memoria caché permite al 80386 funcionar más eficientemente con velocidades menores de DRAM. El paralelismo (pipeline) es una forma especial de manejar el acceso a la memoria para tener un tiempo adicional para direccionar los datos. Un 80386 a 16 MHz permite que dispositivos de memoria con tiempos de acceso de 50 ns o menos operen a toda velocidad. Como es obvio, en la actualidad no hay DRAMs disponibles con estos tiempos de acceso. De hecho, las DRAMs más rápidas actualmente en producción tienen tiempo de acceso de 55 ns o más. Esto significa que se debe encontrar alguna técnica para conectar estos dispositivos de memoria más lentos que el microprocesador. Están disponibles tres técnicas: memoria entrelazada, caché y paralelismo. El 80386 de 16 MHz opera utilizando DRAM de 100 ns en un sistema que utiliza memoria entrelazada.

El paralelismo es la forma preferida para la interface de memoria, porque el microprocesador soporta este tipo de acceso a la memoria. El paralelismo en el 80386 le permite a la memoria un periodo extra del reloj para direccionar los datos. El periodo adicional extiende el tiempo de acceso de 50 ns a 81 ns en un 80386 operando con un reloj de 16 MHz. La *arquitectura paralela* del microprocesador, cuando se recupera (fetch) una instrucción de la memoria, proporciona un tiempo adicional antes que se requiera la siguiente instrucción. Durante este tiempo adicional, la dirección de la siguiente instrucción se envía a través del canal de direcciones. Este tiempo extra (un periodo de reloj) se utiliza para proporcionar un tiempo adicional de acceso para los componentes de memoria más lentos.

No todas las referencias a la memoria pueden aprovechar la arquitectura paralela, lo que significa que algunos ciclos de memoria no se realizan en paralelo con la ejecución de una instrucción. Estos ciclos de memoria no paralelos solicitan un estado de espera, mientras que el ciclo paralelo normal no necesita estados de espera. En general, la arquitectura paralela es una característica que reduce el costo y el tiempo de acceso requerido por la memoria en sistemas de baja velocidad.

No todos los sistemas pueden tomar ventaja de la arquitectura paralela. Estos sistemas son típicamente aquellos que operan a 25 o 33 MHz. En esos sistemas de más velocidad, se debe utilizar otra técnica para incrementar la velocidad del sistema de memoria. El sistema de memoria *caché* mejora el funcionamiento general del sistema de la memoria para aquellos datos que se accesan más de una vez.

Un sistema de memoria caché de alta velocidad se coloca entre el microprocesador y el sistema de la memoria DRAM. Los dispositivos de la memoria caché por lo regular son componentes de memoria TTL con tiempos de acceso menores a 25 ns. En muchos casos vemos sistemas de memoria caché que varían entre 32K bytes y 256K bytes. El tamaño de la memoria caché se determina más por la aplicación que por el microprocesador. Si un programa de escritura es pequeño y solicita pocos datos de la memoria, es benéfico un caché pequeño. Si un programa es grande y solicita grandes bloques de memoria, se recomienda el tamaño más grande posible caché. En muchos casos una caché de 64K mejora la velocidad lo suficiente.

La memoria caché funciona de la siguiente manera. Cada vez que el microprocesador accesa la memoria, el sistema caché primero se prueba para ver si los datos están en él. Si los datos se encuentran en caché, tenemos un *acierto* del caché. Cada vez que ocurre un acierto se obtienen los datos del caché sin ningún estado de espera. Si los datos no se encuentran en el caché, tenemos una *falla* del caché. Cuando ocurre una falla, los datos se leen del DRAM, se almacenan en el caché y de aquí los lee el microprocesador. Esto por supuesto requiere de estados de espera para hacer que la velocidad del microprocesador iguale a la más lenta de la memoria DRAM.

Al escribir datos a la memoria, también se escriben al caché. Aunque esto causa normalmente estados de espera para la DRAM, si los datos se leen posteriormente, ya se encuentran en el caché

lo cual significa una operación de estado de espera de cero en lecturas subsecuentes de los mismos datos. Este método de escribir se llama una operación de *escritura a través del caché*.

En un sistema de memoria caché, los datos se organizan en bloques de bytes. Los bloques tienen un tamaño de 2 a 16 bytes. Cada vez que hay una falla del caché, el microprocesador lee de 2 a 16 bytes de datos de la memoria caché. En el caso del microprocesador 80386, utilizamos un tamaño de bloque de 16 bytes o cuatro localidades de memoria de 32 bits. Cuando los datos se leen (recuperan) de la memoria como localidades de 32 bits, llamamos a la transferencia una *transferencia en ráfaga*. La razón por la cual el caché está organizado de esta forma es porque la mayoría de los programas y datos son secuenciales. Al transferir cuatro dobles palabras de 32 bits desde la memoria al caché por cada falla, en realidad estamos almacenando en el caché los siguientes datos o instrucciones utilizados por el microprocesador. Este método de llenar el caché se llama *adelantar el caché*.

La figura 14-4 muestra un sistema típico de memoria caché de 32K bytes. El caché se organiza como memoria de 8K × 49. Esto significa que hay 8K localidades, y que cada localidad tiene 49 bits. La memoria de 49 bits de ancho se divide en dos secciones: una sección de 32 bits que tiene los datos, la otra es de 17 bits y tiene un marcador. El marcador es una parte de la dirección de la memoria (A32 – A15). Esto significa que hay 32K bytes (8K × 32) para el almacenamiento de datos y 8K × 17 para los marcadores. Esta información nunca se incluye en el tamaño de la memoria caché, así que es un caché de 32K bytes. Los marcadores se almacenan dentro del controlador caché y por lo tanto no se necesita ninguna memoria externa para el almacenamiento de marcadores.

Este es un caché mapeado directamente porque sólo los 17 bits de la dirección de más a la izquierda se almacenan en el campo de un marcador. Los restantes 13 bits de la dirección (A14 – A2) se utilizan para accesar una de 8K localidades de datos (4 bytes) y una localidad en la memoria de marcadores de 8K. (Véase la figura 14-5 para la organización de la memoria caché.)

Cada vez que el microprocesador envía una dirección al sistema de memoria, el controlador caché verificará su marcador (A31 – A15) para determinar si la localidad direccionada por A14 – A2 está almacenada en el caché. La comparación prueba la dirección almacenada en el campo de los marcadores contra los bits de dirección A31 – A15 para verificar si existe una coincidencia o acierto. Si el marcador es igual, el microprocesador obtiene los datos de la memoria caché. Si no es igual, el microprocesador lee los datos de la memoria principal, los almacena en el caché, y también almacena el marcador en la memoria de marcadores. Para leer los datos de la memoria principal tal vez sean necesarios hasta cuatro estados de espera, mientras que una lectura del caché no necesita esperas.

Si los datos se escriben a la memoria, el controlador del caché escribe los datos a una localidad del caché y también a la memoria de marcadores. Una escritura normalmente necesita un estado de espera.

Supongamos que el microprocesador acaba de leer datos de la localidad 01007FF0H de la memoria. Los datos de esta localidad se almacenan en la memoria caché en la localidad 111 1111 1111 00XX (13 bits menos significativos de la dirección). El marcador que se almacena en esta memoria para este acceso es 0000 0001 0000 0000 0. Si ésta es la misma dirección almacenada actualmente en la localidad de la memoria del marcador, tenemos un acierto.

Si tenemos un acierto en este ejemplo para la localidad 01007FF0H y el microprocesador ahora intenta leer la localidad 02007FF0H, direccionamos la misma localidad en el caché y en los marcadores. Debido a que los 17 bits más a la izquierda de la dirección no son iguales, tenemos

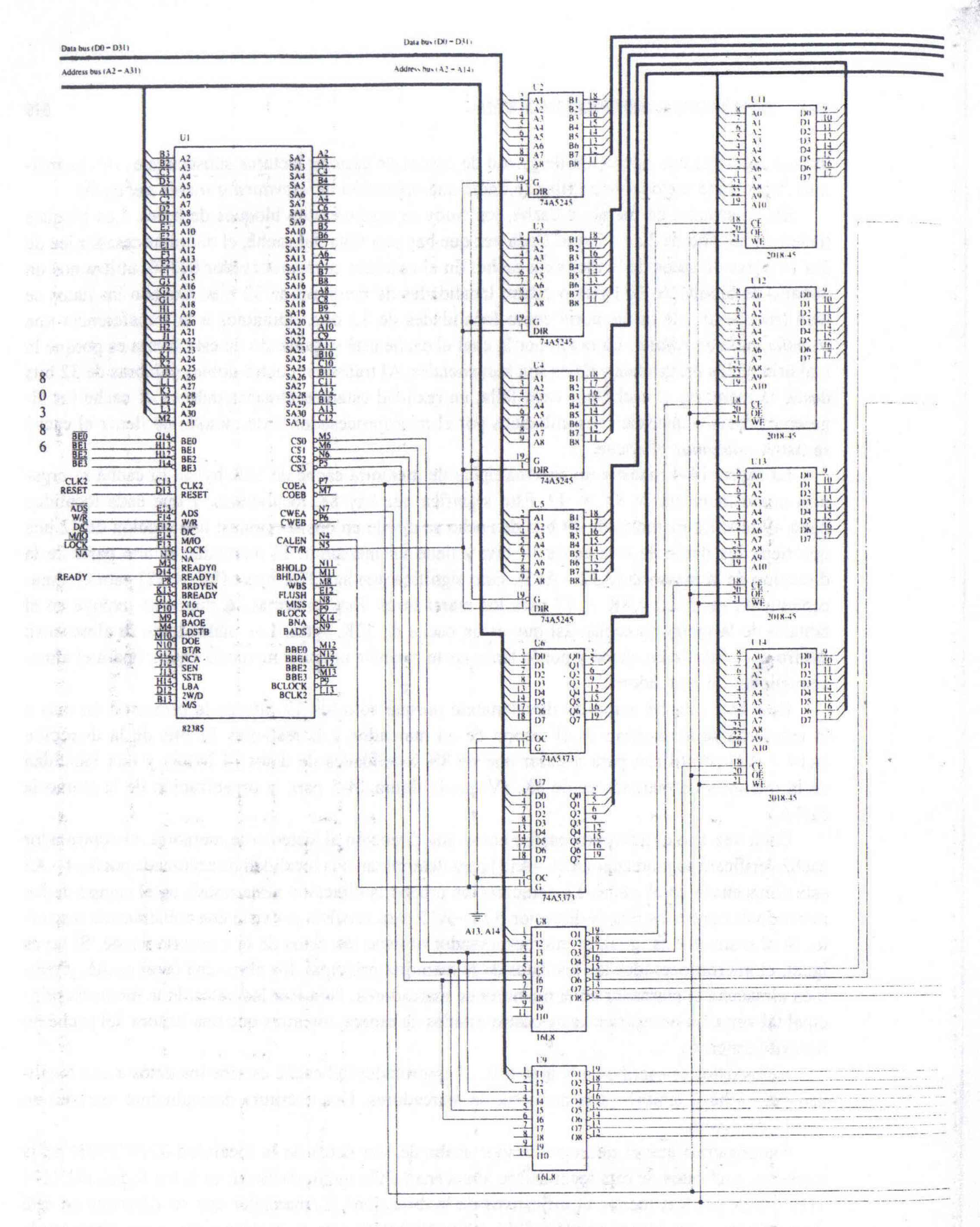
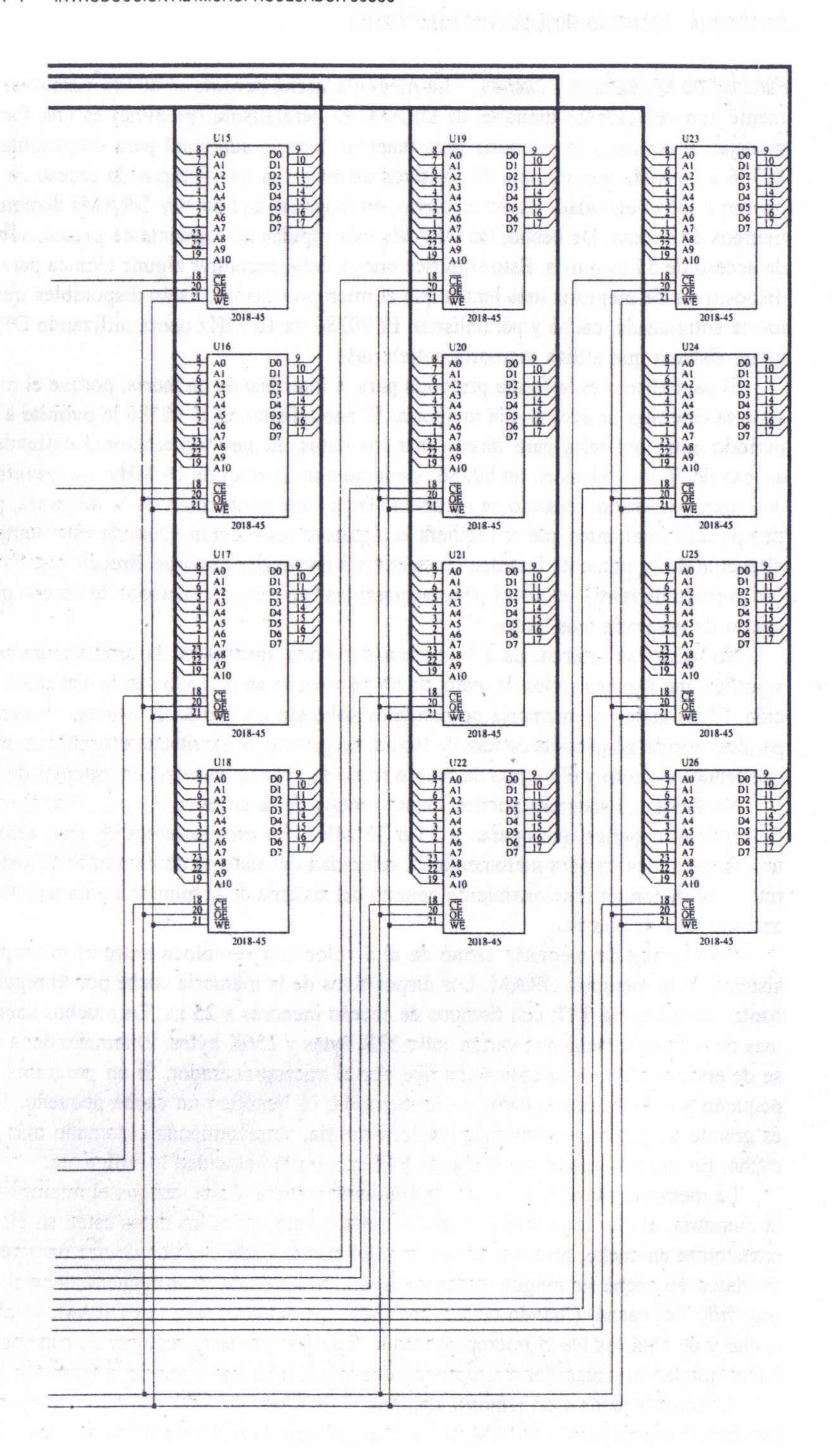


FIGURA 14–4 Un sistema de memoria caché de 32K bytes manejada por un controlador caché 82385. La memoria consiste en una pila de RAM de alta velocidad de 2K x 8 que tiene un tiempo de acceso de 45 ns.


U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10
CE OE WE
D0 D1 D2 D3 D4 D5 D6 D7
2018-45

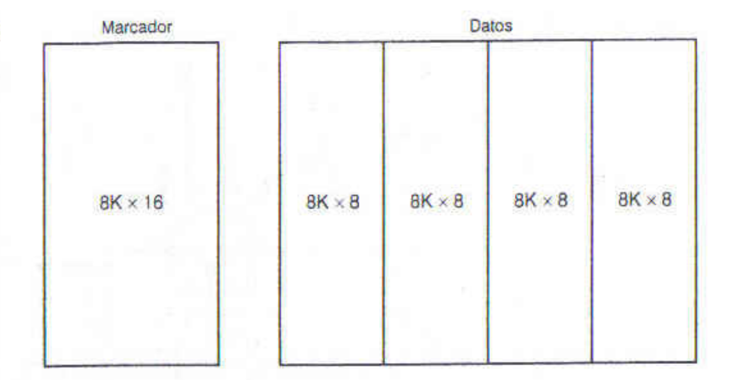


FIGURA 14–5 Organización de la memoria caché de 32K bytes mapeada directamente.

una falla. Por suerte, los accesos de la memoria casi siempre son secuenciales así que este tipo de falla rara vez sucede. Es más común que la siguiente falla ocurra al direccionar la dirección 0100FFF0H. (Observe que 0100FFF0H está 32K bytes arriba de 01007FF0H.)

Los sistemas de memoria entrelazada permiten que los tiempos de acceso de la memoria puedan ser más largos sin la necesidad de estados de espera. Un sistema de memoria entrelazada requiere de dos conjuntos completos de canales y un controlador que proporcione las direcciones para cada canal.

Una memoria entrelazada se divide en dos partes. Para el microprocesador 80386, una parte contiene las direcciones de 32 bits 000000H – 000003H, 000008H – 00000BH, etcétera, mientras que la otra parte contiene las direcciones 000004 – 000007, 00000CH – 00000FH, etcétera. En tanto el microprocesador accesa las direcciones 000000H – 000003H, la lógica de control entrelazado genera la señal de habilitación de dirección para las localidades 000004H – 000007H. Este proceso continúa mientras el microprocesador accese las localidades consecutivas de memoria. Este proceso para direccionar los bancos de memoria alternos, alarga la cantidad del tiempo de acceso proporcionado a la memoria porque la dirección se genera antes de que el microprocesador necesite los datos. Esto se debe a la arquitectura paralela del microprocesador ya que envía la siguiente dirección antes que se lean los datos de la última dirección.

El problema del entrelazado, aunque no es mayor, es que las direcciones de la memoria deben ser accesadas para que cada sección se direccione alternamente. Esto no siempre sucede conforme se ejecuta un programa. Bajo una ejecución normal del programa, el microprocesador accesa la memoria alternamente aproximadamente el 93% del tiempo. El restante 7%, el microprocesador direcciona datos en la misma sección de memoria, lo que significa que en este 7% de los accesos de la memoria, el sistema de la memoria tiene que causar estados de espera debido al tiempo de acceso reducido. El tiempo de acceso se reduce porque la memoria tiene que esperar hasta que los datos anteriores sean transferidos antes de poder obtener su dirección. Esto lo deja con menos tiempo de acceso; por lo tanto se requiere de un estado de espera para los accesos en el mismo banco de memoria.

Vea la figura 14-6 donde se presenta el diagrama de temporización del direccionamiento como aparece en las terminales de dirección del microprocesador. Este diagrama de tiempo muestra cómo se le da salida a la siguiente dirección antes que se accesen los datos actuales. También muestra cómo el tiempo de acceso se incrementa utilizando las direcciones de la memoria entre-

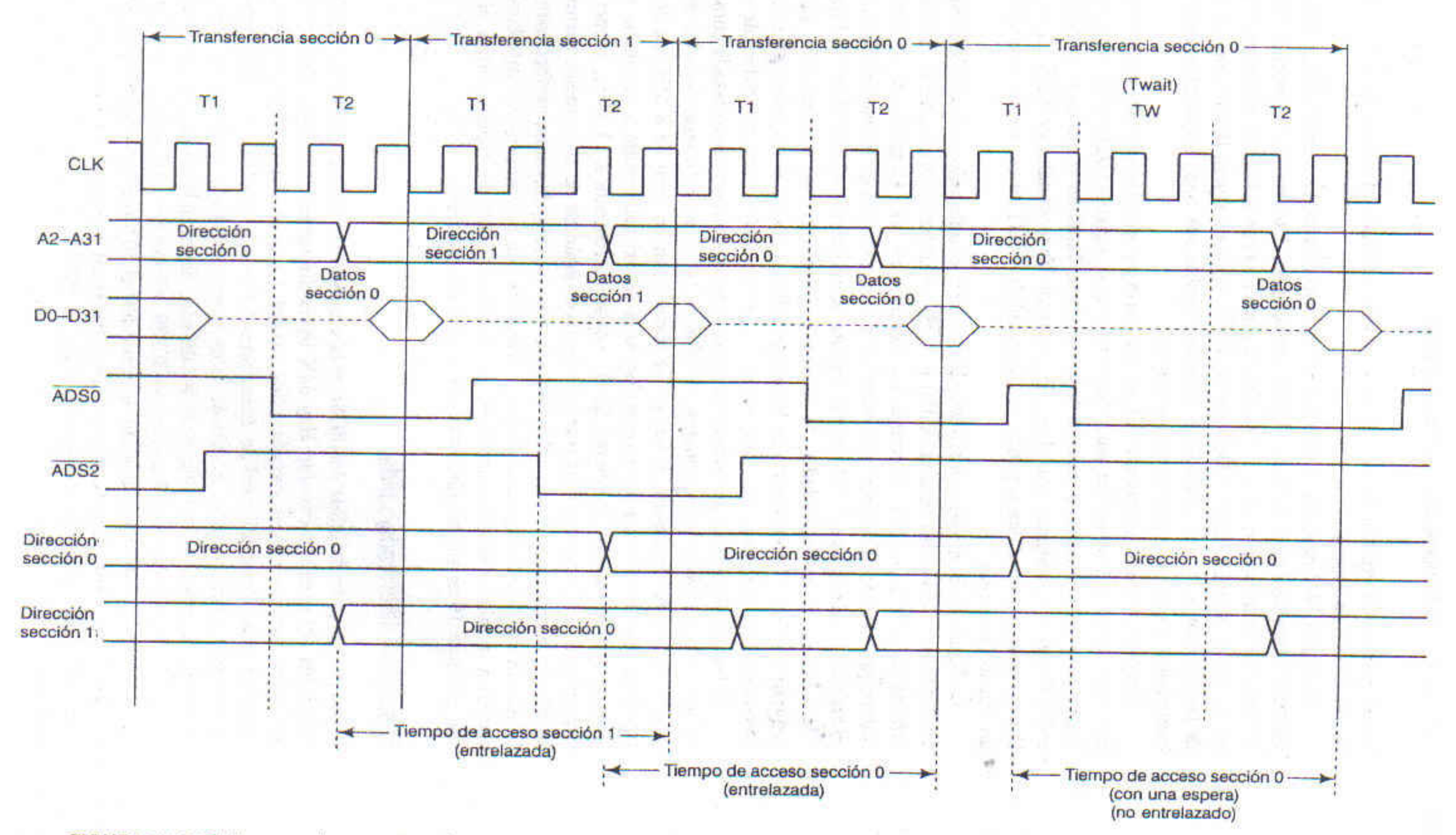


FIGURA 14–6 El diagrama de temporización de un sistema de memoria entrelazada mostrando los tiempos de acceso y las señales de direccionamiento para ambas secciones de la memoria.

lazada para cada sección de la memoria, comparados con el acceso no entrelazado, que requiere de un estado de espera.

La figura 14-7 muestra el controlador de entrelazado. Es cierto que éste es un circuito lógico bastante complejo que necesita alguna explicación. Primero, si la entrada SEL (utilizada para seleccionar esta sección de la memoria) está inactiva (un 0 lógico), entonces la señal de $\overline{WAIT}$ es un 1 lógico. Además, tanto $\overline{ADS0}$ como $\overline{ADS1}$, utilizadas para la habilitación de estroboscópica de la dirección a las secciones de la memoria, ambas son unos lógicos haciendo que los registros conectados a ellas se vuelvan transparentes.

Tan pronto como la entrada SEL se convierta en un 1 lógico, este circuito empieza a funcionar. La entrada A3 se utiliza para determinar qué registro (U2B o U5A) se convierte en un 0 lógico para seleccionar una sección de la memoria. También la terminal de $\overline{ADS}$ que se convierte en un 0 lógico se compara con el estado previo de las terminales de $\overline{ADS}$. Si la misma sección de la memoria se accesa una segunda vez, la señal de $\overline{WAIT}$ se convierte en un 0 lógico solicitando un estado de espera.

La figura 14-8 muestra un sistema de memoria entrelazado, que utiliza el circuito de la figura 14-7. Observe cómo las señales $\overline{ADS0}$ y $\overline{ADS1}$ se utilizan para capturar la dirección para una u otra de las secciones de la memoria. La memoria en cada banco es de 16 bits de ancho. Si los accesos a la memoria requieren de datos de 8 bits, entonces en la mayoría de los casos el sistema causa estados de espera. Conforme se ejecuta un programa, el 80386 obtiene instrucciones de 16 bits a la vez de las localidades de la memoria normalmente secuenciales. La ejecución de un programa utiliza el entrelazado en la mayoría de los casos. Si un sistema va a direccionar sobre datos de 8 bits, no es seguro que la memoria entrelazada reduzca el número de estados de espera.

El tiempo de acceso permitido por un sistema de memoria entrelazada como el que se muestra en la figura 14-8, se incrementa de 78 ns a 145.5 ns utilizando un reloj de 16 MHz. (Si se inserta un estado de espera, el tiempo de acceso con un reloj de 8 MHz es de 140.5 ns, lo que significa que un sistema en niveles funciona aproximadamente a la misma velocidad que un sistema con un estado de espera.) Si el reloj se incrementa a 33 MHz, la memoria entrelazada requiere de 72.75 ns, mientras que las interfaces estándar de la memoria permiten 46 ns para el acceso de la memoria. A esta velocidad más alta del reloj funciona perfectamente una DRAM de 55 ns, sin estados de espera cuando las direcciones de la memoria son entrelazadas. Si ocurre un acceso a la misma sección, entonces se inserta un estado de espera porque el microprocesador sólo permite 39 ns sin ningún estado de espera en este caso.

El sistema de entrada/salida

El sistema de entrada/salida del 80386 es básicamente el mismo que se encuentra en las versiones anteriores del microprocesador. Hay 64K bytes diferentes de espacio de entrada/salida disponibles si se implementa una entrada/salida aislada. La dirección del puerto de entrada/salida aparece en las conexiones del canal de direcciones A15 – A2, con $\overline{BE3}$ – $\overline{BE0}$ utilizados para seleccionar un byte, palabra o doble palabra de datos de entrada/salida. Si se utiliza entrada/salida mapeada en memoria, entonces el número de localidades de entrada/salida puede ser cualquier cantidad hasta 4G bytes. Casi todos los sistemas 80386 utilizan entradas/salidas aisladas porque el esquema de protección de entrada/salida utilizado por el 80386 es en el modo protegido.

La figura 14-9 muestra el mapa de entrada/salida para el microprocesador 80386. A diferencia del mapa de entrada/salida del 8086, el 80386 utiliza un sistema de entrada/salida completo de 32 bits de ancho dividido en cuatro bancos, ya que el sistema de la memoria está dividido

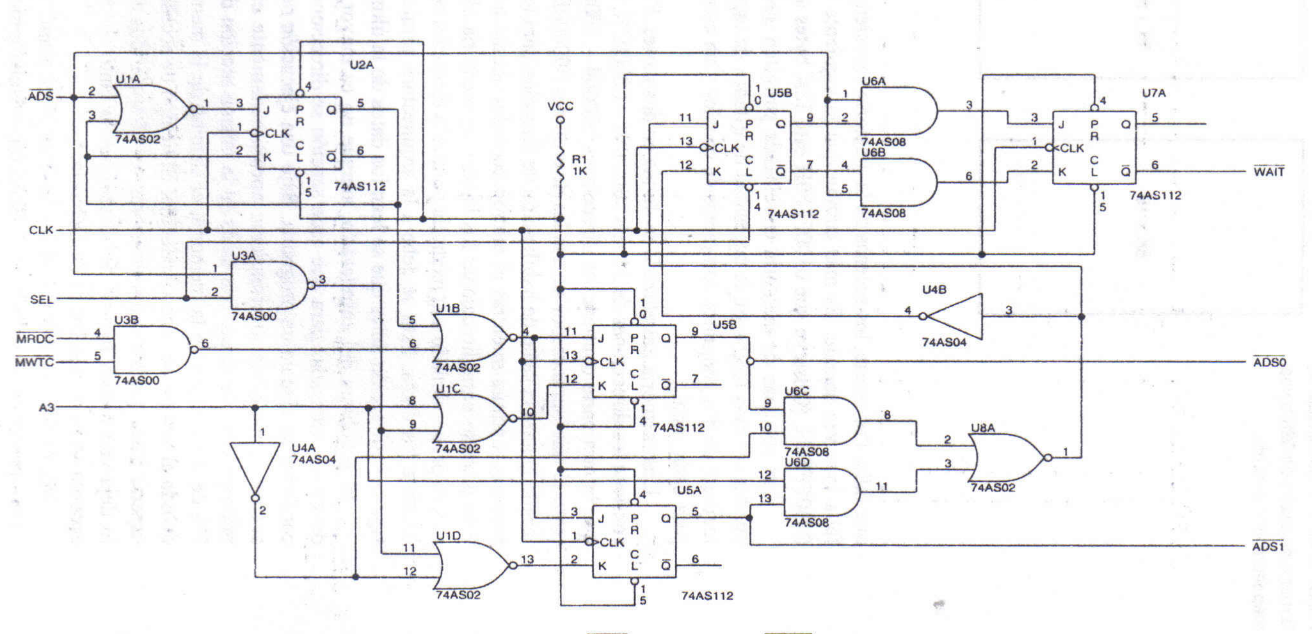


FIGURA 14–7 Lógica de control que genera las señales separadas de $\overline{\text{ADS}}$ y una señal de $\overline{\text{WAIT}}$ utilizada para controlar la memoria entrelazada en niveles.

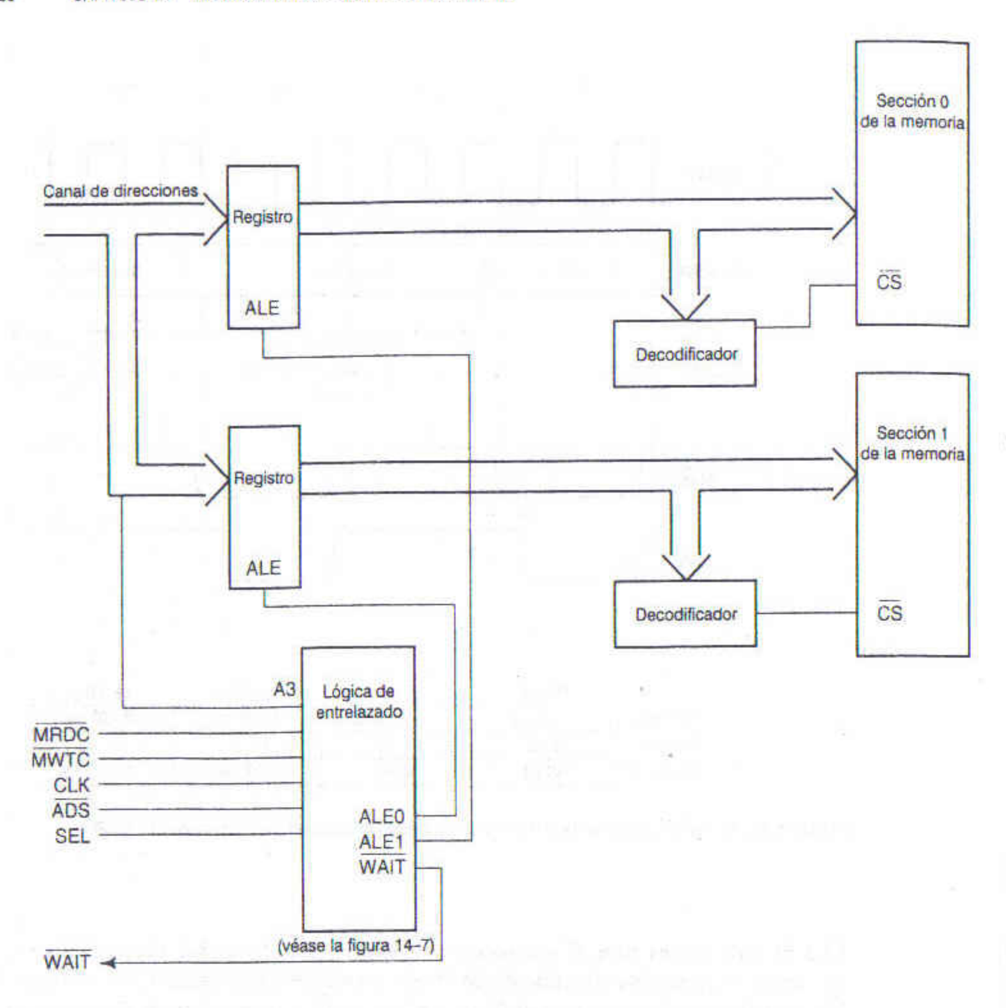


FIGURA 14–8 Un sistema de memoria entrelazada mostrando los registros de la dirección y el circuito lógico para el entrelazado.

en cuatro bancos. La mayor parte de las transferencias de entrada/salida son de 8 bits de ancho porque a menudo se utiliza código ASCII (un código de 7 bits) para la transferencia de datos alfanuméricos entre el microprocesador e impresoras y teclados. Recientemente, los periféricos de entrada/salida que son de 16 y hasta de 32 bits de ancho (EISA o canal local) han aparecido para los sistemas tales como la memoria de disco e interfaces de video. Estas trayectorias de entrada/salida más anchas incrementan la velocidad de la transferencia de datos entre el microprocesador y los dispositivos de entrada/salida comparados a las transferencias de 8 bits.

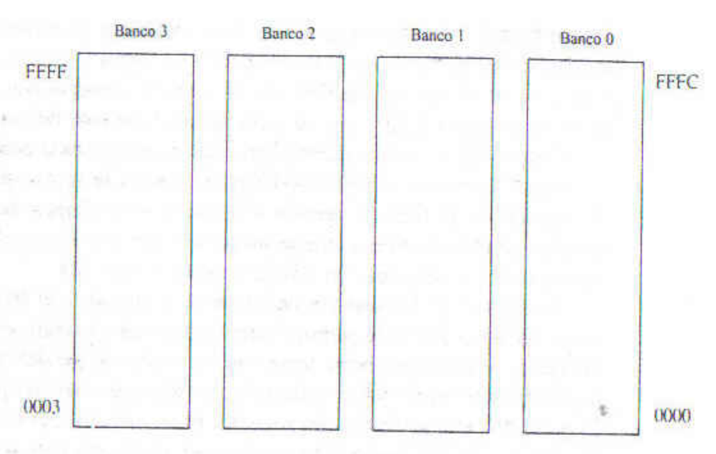


FIGURA 14-9 Mapeo aislado de entrada/salida para el microprocesador 80386. Aquí se utilizan cuatro bancos de 8 bits cada uno para direccionar 64K diferentes localidades de entrada/salida. La entrada/salida está numerada de la localidad 0000H a FFFFH.

Las localidades de entrada/salida están numeradas de 0000H a FFFFH. Una parte del mapa de entrada/salida está asignada para el coprocesador aritmético 80387. Aunque los números de puerto para el coprocesador están muy por encima del mapa de entrada/salida normal, es importante que se tomen en cuenta al decodificar el espacio de entrada/salida para evitar traslapes. El coprocesador utiliza la localidad de entrada 800000F8H – 800000FFH para las comunicaciones entre el 80387 y 80386. (Recuerde que el 80287 utilizó las direcciones de entrada/salida 00F8H – 00FFH para el mismo propósito.) Debido a que frecuentemente decodificamos sólo las conexiones de dirección A15 – A2 para seleccionar un periférico de entrada/salida, hay que tomar en cuenta que el coprocesador activará los dispositivos en las localidades 00F8H – 00FFH a no ser que la línea de dirección A31 también sea decodificada.

La única nueva característica agregada al 80386 en relación a la entrada/salida es la información privilegiada de entrada salida agregada a la cola final del TSS cuando el 80386 se opera en modo protegido. En el modo protegido, como se describe en la sección sobre el manejo de la memoria del 80386, una localidad de entrada/salida se puede bloquear o inhibir. Si se direcciona la localidad bloqueada de entrada/salida, se genera una interrupción (tipo 13). Este esquema se agrega para que el acceso de entrada/salida pueda ser prohibido en un ambiente multiusuario. El bloqueo es una extensión de la operación de modo protegido, así como lo son los niveles de privilegio.

Señales de control de memoria y de entrada/salida

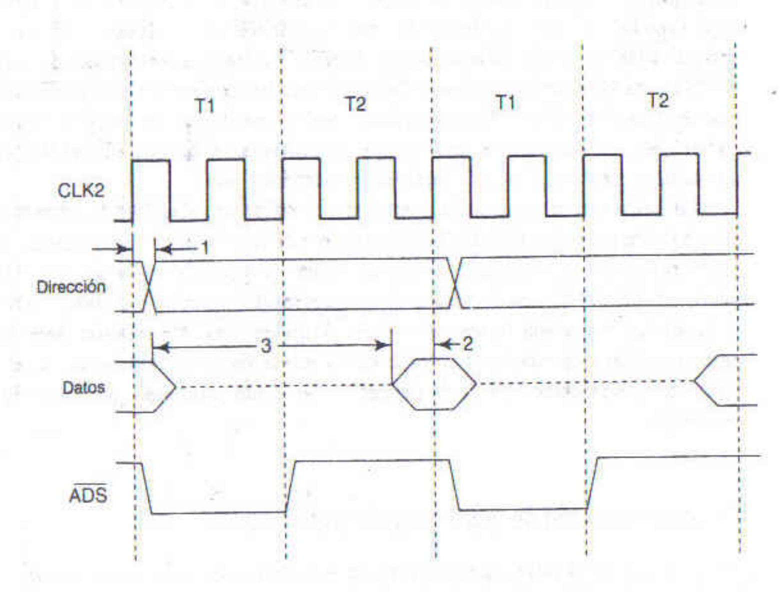
Al igual que en el 8086, la memoria y la entrada/salida están controlados por señales separadas. La señal de $M/\overline{IO}$ indica si la transferencia es en memoria o en entrada/salida. Además de la $M/\overline{IO}$, los sistemas de memoria y de entrada/salida deben leer o escribir datos. La señal de $W/\overline{R}$ es un 0 lógico para una operación de lectura, y un 1 lógico para una operación de escritura. La señal $\overline{ADS}$ se utiliza para habilitar estas dos señales de control. Esta es una desviación del sistema 8086 que utiliza señales separadas para la lectura y escritura de memoria y para la lectura y escritura de entrada/salida.

Temporización. La temporización es muy importante para comprender la interface de memoria y de entrada/salida al microprocesador 80386. La figura 14-10 muestra el diagrama de temporización para un ciclo de lectura a la memoria no paralelo. Observe que la temporización está referida a la señal de entrada CLK2 y que un ciclo de canal consiste de cuatro periodos del reloj.

Cada ciclo de canal contiene 2 estados de reloj y cada estado (T1 y T2) contiene 2 periodos del reloj. Observe en la figura 14-10 que el tiempo de acceso se indica como el tiempo número 3. La versión de 16 MHz le permite a la memoria un tiempo de acceso de 78 ns antes de que se inserten estados de espera en este modo de operación sin paralelismo. Para seleccionar el modo sin paralelismo, se coloca un 1 lógico en la terminal $\overline{NA}$.

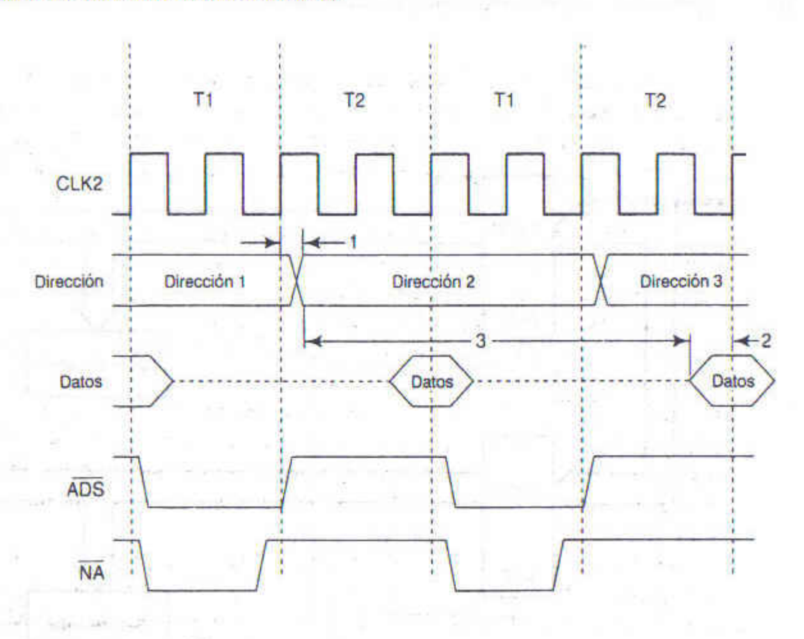
La figura 14-11 muestra el tiempo de lectura cuando el 80386 se opera en el modo de paralelismo. Observe que se le permite tiempo adicional a la memoria para accesar los datos porque la dirección se envía en forma temprana. El modo de paralelismo se selecciona colocando un 0 lógico en la terminal $\overline{NA}$ y utilizando registros transparentes para capturar la dirección paralela. El pulso del reloj aplicado a los registros transparentes es la señal $\overline{ADS}$. Los registros transparentes para las direcciones se deben utilizar en un sistema con paralelismo así como con los bancos de memoria entrelazada. El número mínimo de bancos entrelazados es de dos, y se han utilizado cuatro con éxito en algunas aplicaciones.

Observe que en la dirección "paralela" aparece un estado de reloj completo antes de lo que aparece normalmente en el acceso no paralelo. En la versión de 16 MHz del 80386, esto permite



	33 MHz	25 MHz	20 MHz	16 MHz
Tiempo 1:	4–15 ns	4–21 ns	4–30 ns	4–36 ns
Tiempo 2:	5 ns	7 ns	11 ns	11 ns
Tiempo 3:	46 ns	52 ns	59 ns	78 ns

FIGURA 14–10 Temporización de la lectura sin paralelismo para el microprocesador 80386.



	33 MHz	25 MHz	20 MHz	16 MHz
Tiempo 1:	4–15 ns	4–21 ns	4–30 ns	4–36 ns
Tiempo 2:	5 ns	7 ns	11 ns	11 ns
Tiempo 3:	80 ns	92 ns	109 ns	145.5 ns

FIGURA 14–11 Temporización de la lectura con paralelismo para el microprocesador 80386.

67.5 ns adicionales para el acceso de la memoria. En el sistema sin paralelismo, se tenía un tiempo de acceso para la memoria de 78 ns y en el sistema con paralelismo se tienen 145.5 ns. Las ventajas del sistema con paralelismo son que no se requieren estados de espera (en muchos, pero no en todos los ciclos del canal) y que se pueden conectar al microprocesador dispositivos de memoria mucho más lentas. La desventaja es que se necesita una memoria entrelazada para utilizar el paralelismo, que necesita de circuitos adicionales y de estados de espera ocasionales.

Estados de espera

Así como con el 8086, necesitamos introducir estados de espera si los tiempos para el acceso de la memoria son largos, comparados con el tiempo permitido por el 80386 para el acceso de memoria. En un sistema sin paralelismo de 33 MHz, el tiempo de acceso es sólo de 46 ns. No existe ninguna memoria DRAM que tenga un tiempo de acceso de 46 ns. Esto significa que se tienen que introducir estados de espera para accesar DRAM (1 espera para DRAM de 60 ns) o EPROM que tiene un tiempo de acceso de 100 ns (2 esperas).

La entrada de $\overline{\text{READY}}$ controla si se insertan o no estados de espera en el ciclo del canal. La entrada $\overline{\text{READY}}$ del 80386 es una entrada dinámica que se tiene que activar durante cada ciclo del canal. La figura 14-12 muestra unos cuantos ciclos de canal con un ciclo normal (0 espera) y uno que contiene un solo estado de espera. Observe cómo se controla $\overline{\text{READY}}$ para causar 0 o 1 espera.

La señal de $\overline{\text{READY}}$ será probada al final de un ciclo de canal para determinar si el ciclo del reloj es T2 o TW. Si en este momento $\overline{\text{READY}}$ = 0, es el final del ciclo de canal o T2. Si $\overline{\text{READY}}$ es 1 al final de un ciclo de reloj, el ciclo es TW y el microprocesador continúa probando $\overline{\text{READY}}$ buscando una lógica de 0 y el final del ciclo de canal.

En el sistema sin paralelismo, cada vez que $\overline{\text{ADS}}$ se convierte en un cero lógico, $\overline{\text{READY}}$ = 1. Después de que $\overline{\text{ADS}}$ vuelve a un 1 lógico, los flancos positivos del reloj se cuentan para generar la señal de $\overline{\text{READY}}$. La señal $\overline{\text{READY}}$ se convierte en un 0 lógico después del primer ciclo de reloj para insertar 0 estados de espera. Si se inserta un estado de espera, la línea $\overline{\text{READY}}$ tiene que permanecer como un 1 lógico hasta que hayan pasado por lo menos dos ciclos. Si se desean estados de espera adicionales, entonces tiene que pasar tiempo adicional antes de que $\overline{\text{READY}}$ sea cero.

La figura 14-13 muestra un circuito que inserta de 0 a 3 estados de espera para varios accesos de la memoria. En el ejemplo, un estado de espera se produce para un acceso a DRAM y 2 esperas para un acceso a EPROM. El 74F164 se reinicia (limpia) cada vez que $\overline{\text{ADS}}$ está en nivel bajo y D/$\overline{\text{C}}$ está en alto. Empieza a cambiar después que $\overline{\text{ADS}}$ regresa al nivel de 1 lógico. Conforme cambia, el 00000000 en el registro de corrimiento empieza a llenarse con unos lógicos desde la conexión QA a la conexión QH. Las cuatro salidas diferentes están conectadas a un multiplexor inversor que genera la señal $\overline{\text{READY}}$ activa en bajo.

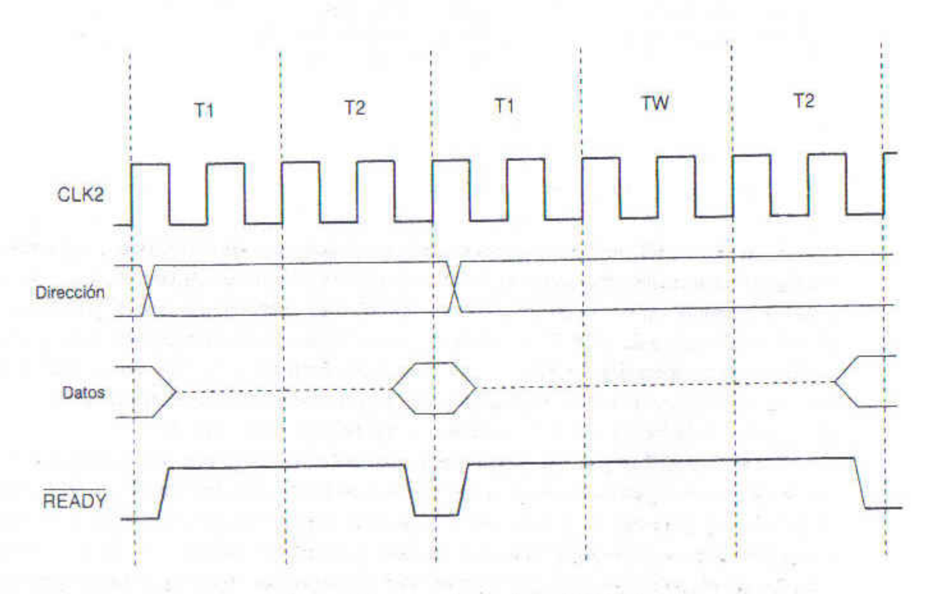


FIGURA 14–12 Se muestra un 80386 sin paralelismo con 0 y 1 estados de espera.

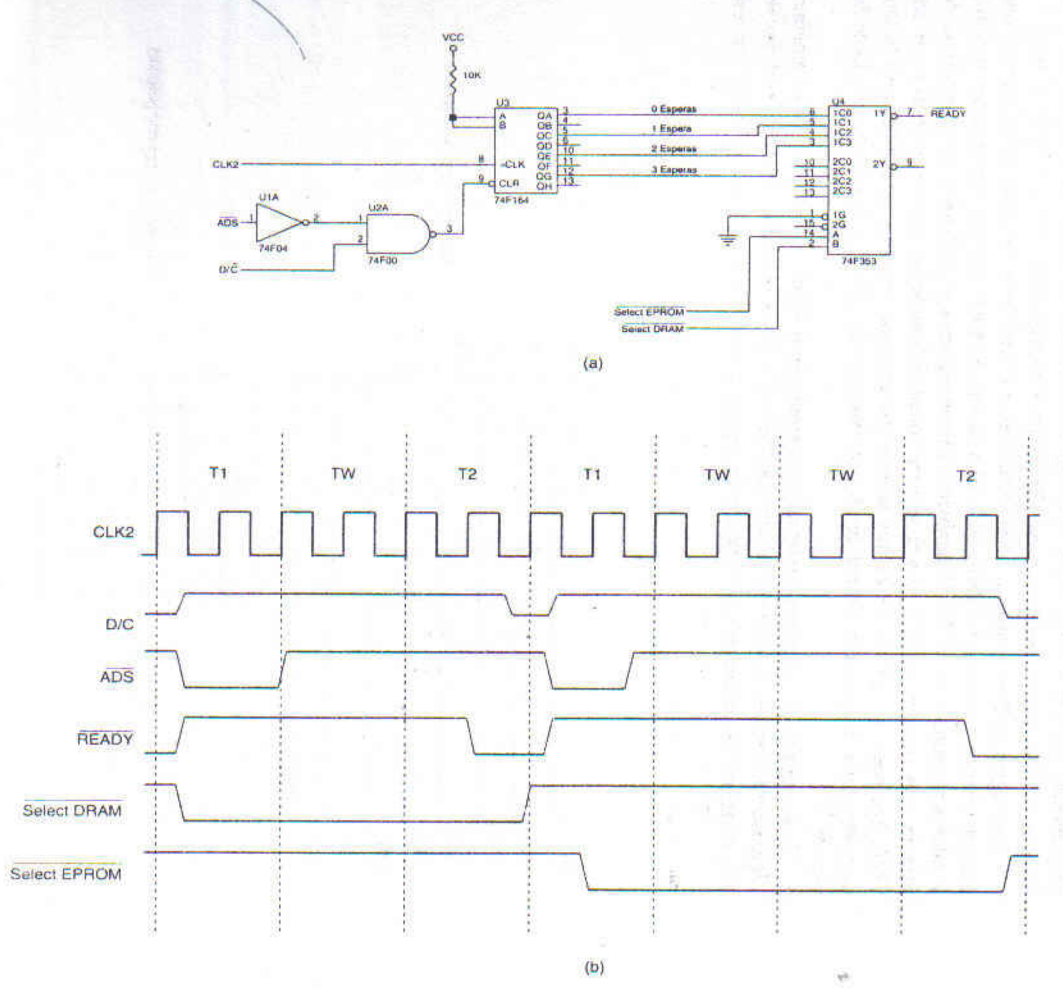


FIGURA 14–13 Circuito (a) y temporización (b) que selecciona un estado de espera para DRAM y 2 para EPROM.

14–2 LA ESTRUCTURA DE REGISTROS DEL 80386

La estructura de registros del 80386 es una versión muy expandida de los registros que se encuentran en el microprocesador 80286. La figura 14-14 muestra la estructura de registros visible al programa del microprocesador 80386. Los registros se dividen en las secciones exactamente como están en el 8086: (1) *propósito general*, (2) *registros de segmento (selector)*, y (3) *Control*.

Los registros de propósito general se accesan a través de la mayoría de las instrucciones y están diseñados para contener datos de 8, 16 o 32 bits. El 8086 contiene registros de tamaño byte y palabra, mientras que el 80386 también contiene registros extendidos o con tamaño de doble palabra. Los registros de 8 y 16 bits se direccionan utilizando los mismos nombres que con el 8086. Los nuevos registros de 32 bits se direccionan como registros extendidos con la letra E aumentando la designación normal de 16 bits. Por ejemplo, EAX es la versión extendida de 32 bits del AX.

Los registros de segmento (selector) son similares al 8086, excepto que éste contiene dos registros adicionales de segmento etiquetados FS y GS. En la operación en modo real, los registros de segmento contienen una dirección de segmento exactamente como en el microprocesador

EAX		AH	AL	AX	
EBX		BH	BL	BX	
ECX		CH	CL	CX	
EDX		DH	DL	DX	Registros de propósito general
ESP		SP			
EBP		BP			
EDI		DI			
ESI		SI			

CS	
DS	
ES	Registros de segmento (selector)
SS	
FS	
GS	

EIP		IP	Registros de control
EFLAGS		FLAGS	

FIGURA 14–14 Estructura interna del microprocesador 80386 ilustrando los registros de segmento, de propósito general y de conteo.

8086 y en la operación de modo protegido contienen un selector. Se utiliza un prefijo para direccionar datos utilizando FS o GS. Por ejemplo, para utilizar GS para direccionar datos en el segmento GS encontramos la instrucción MOV EAX,GS:DATO. El prefijo GS: o FS: se utiliza para seleccionar estos dos nuevos registros de segmento para cualquier modo de direccionamiento.

Los registros de control consisten de un apuntador de instrucciones (EIP) de 32 bits y un registro de bandera (EFLAGS) de 32 bits. Durante la operación en modo real, sólo los 16 bits más a la derecha del EIP contienen la dirección del desplazamiento de la siguiente instrucción que ejecutará el programa. En la operación en modo protegido, todos los 32 bits del EIP se utilizan para accesar segmentos que pueden contener un programa o datos que pueden llegar hasta 4G de bytes. Esto es una mejora significativa sobre los segmentos de 64K bytes disponibles en el 80286.

EJEMPLO 14–1

```
                              .386

                              ;ejemplo de instrucciones del 80386 utilizando una variedad de
                              ;modos de direccionamiento

0000                          PROGRAM  SEGMENT  USE16

0000 66| B8 22223333                   MOV    EAX,22223333H
0006 66| BB 44445555                   MOV    EBX,44445555H
000C BF 1000                           MOV    DI,1000H
000F BD 2000                           MOV    BP,2000H
0012 8A 0D                             MOV    CL,[DI]
0014 3E: 8A 6E 00                      MOV    CH,DS:[BP]
0018 66| 03 D8                         ADD    EBX,EAX
001B 66| 8B F3                         MOV    ESI,EBX
001E 66| F7 E6                         MUL    ESI
0021 67| 66| 89 01                     MOV    [ECX],EAX
0025 64: 67| 88 0403                   MOV    FS:[EBX+EAX],AL
002A 65: 67| 8B 07                     MOV    AX,GS:[EDI]

002E                          PROGRAM  ENDS

                                       END
```

El ejemplo 14-1 indica una muestra de las instrucciones que utilizan los registros de 32 bits. Cualquier instrucción operando en modo real o protegido puede utilizar cualquier registro de 32 bits. El ejemplo empieza con un .386 que le indica al ensamblador que el programa es para un 80386. El directivo de USE 16 le indica al ensamblador utilizar direcciones de desplazamiento de 16 bits para la operación en modo real. En el modo protegido con frecuencia incluimos el directivo de USE 32 que le indica al ensamblador utilizar direcciones de desplazamiento de 32 bits. En el apéndice A se indica más sobre los directivos de USE 16 o USE 32.

El registro EFLAG se muestra en la figura 14-15. Observe que los 16 bits más a la derecha son idénticos al registro de banderas del 80286. Los bits de bandera de RF y VM son nuevos para el 80386. RF (bandera de resume) se utiliza en la depuración de programas a fin de desactivar temporalmente algunas interrupciones durante la siguiente instrucción. El bit de bandera VM (modo virtual) selecciona el modo virtual 8086 mientras que el 80386 se opera en el modo protegido. Este modo especial se examina en una sección posterior de este capítulo.

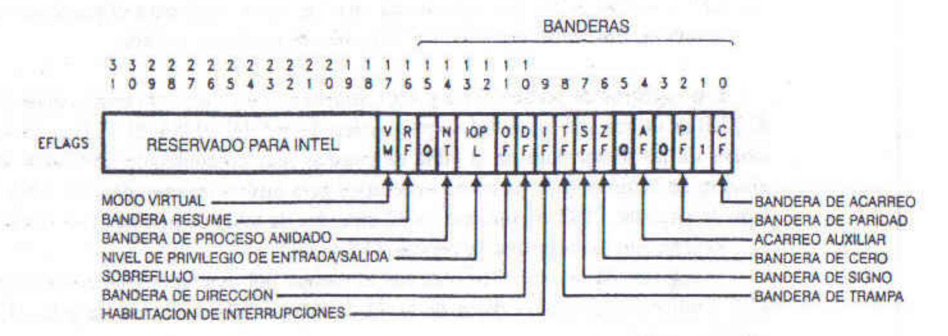


FIGURA 14–15 El registro EFLAG. (Por cortesía de la Corporación Intel.)

Registros de direccionamiento del sistema

Los registros de direccionamiento del sistema son: GDTR (Registro de tabla de descriptores globales), IDTR (registro de tabla de descriptores de interrupciones), y TR (registro de tareas). Todos los registros de direccionamiento del sistema son registros invisibles al programa que no se accesan directamente, excepto para cargar o almacenar. Los primeros tres se utilizan para direccionar áreas de la memoria que contienen las tablas de descriptores globales, locales y de interrupciones, que se utilizan durante la operación del modo protegido. El registro de tarea (TR) se utiliza para accesar un selector que define la tarea actual en la forma de TSS (segmento de estado de la tarea). Estos registros realizan las mismas funciones que en el microprocesador 80286 excepto que la dirección base es de 32 bits de ancho en lugar de 24 bits y el límite es de 20 bits en lugar de 16 bits.

Registros de control

Además de EFLAGS y EIP que se describieron antes, existen otros registros de control en el 80386. El registro de control 0 (CR0) es idéntico al MSW (palabra de estado de la máquina) que se encuentra en el microprocesador 80286, excepto que tiene 32 bits de ancho en lugar de 16 bits. Los registros de control adicionales son CR1, CR2 y CR3.

La figura 14-16 muestra la estructura del registro de control del 80386. El registro de control CR1 no se utiliza en el 80386, pero se reserva para productos futuros. El registro de control CR2 contiene la dirección lineal de la última página accesada antes de una interrupción por falla de página. Por último, el registro de control CR3 contiene la dirección base del directorio de páginas. Los 12 bits más a la derecha de la dirección de 32 bits de la tabla de páginas contienen ceros y se combinan con el resto del registro para localizar el inicio de la tabla de páginas de 4K de longitud.

El registro CR0 contiene un número de bits de control especial que se definen como sigue en el 80386:

1. PG — selecciona la página de la tabla de conversión de direcciones lineales a direcciones físicas cuando PG = 1. La página de la tabla de conversión permite que cualquier dirección lineal sea asignada a cualquier localidad de la memoria física.

FIGURA 14-16 Estructura del registro de control del microprocesador 80386.

	MSW							
PG	000000000000000	00000000000	ET	TS	EM	MP	PE	CR0
No utilizado								CR1
Dirección lineal de página antes de una falla								CR2
Base para el directorio de páginas		000000000000						CR3

2. ET — selecciona el coprocesador 80287 cuando ET = 0 o el coprocesador 80387 cuando ET = 1. Este bit se instaló porque cuando el 80386 apareció por primera vez, no estaba disponible el 80387. En la mayoría de los sistemas, ET se activa (1) para indicar que está presente un 80387 en el sistema.
3. TS — indica que el 80386 tiene conmutación de tareas. Si TS = 1, una instrucción del coprocesador numérico causa una interrupción de tipo 7 (coprocesador no disponible).
4. EM — se activa para causar una interrupción de tipo 7 para cada instrucción de ESC. (Las instrucciones ESC se utilizan para codificar instrucciones para el coprocesador 80387.) Frecuentemente se utiliza esta interrupción para emular, con programa, la función del coprocesador. La emulación reduce el costo del sistema, pero requiere de por lo menos 100 veces más tiempo para ejecutar las instrucciones emulando el coprocesador.
5. MP — se activa para indicar que el coprocesador aritmético está presente en el sistema.
6. PE — se activa para seleccionar el modo de operación protegido para el 80386. También se puede desactivar para volver al modo real. Este bit sólo se puede activar en el 80286, ya que para regresar al modo real necesita una reinicialización por circuito, que excluye su uso en la mayoría de los sistemas que utilizan el modo protegido.

Registros de prueba y depuración

Una nueva serie de registros que no se encuentran en el 8086 aparecen en el 80386 como registros de prueba y depuración. Los registros DR0-DR7 facilitan la depuración de programas y los registros TR6 y TR7 se utilizan para probar la paginación y el caché.

La figura 14-17 muestra el conjunto de registros de prueba y depuración. Los primeros cuatro registros de depuración contienen direcciones lineales de 32 bits de puntos de ruptura. (Una *dirección lineal* es una dirección de 32 bits generada por una instrucción de microprocesador que puede o no ser la misma que la dirección física.) Las direcciones de los puntos de ruptura, que pueden localizar una instrucción o dato, constantemente se comparan con las direcciones generadas por el programa. Si ocurre una coincidencia, el 80386 causará, si DR6 y DR7 lo indican, que ocurra una interrupción de tipo 1 (interrupción de trampa o depuración). Estas direcciones de punto de ruptura son muy útiles para la depuración de programas. Los bits de control DR6 y DR7 se definen como sigue:

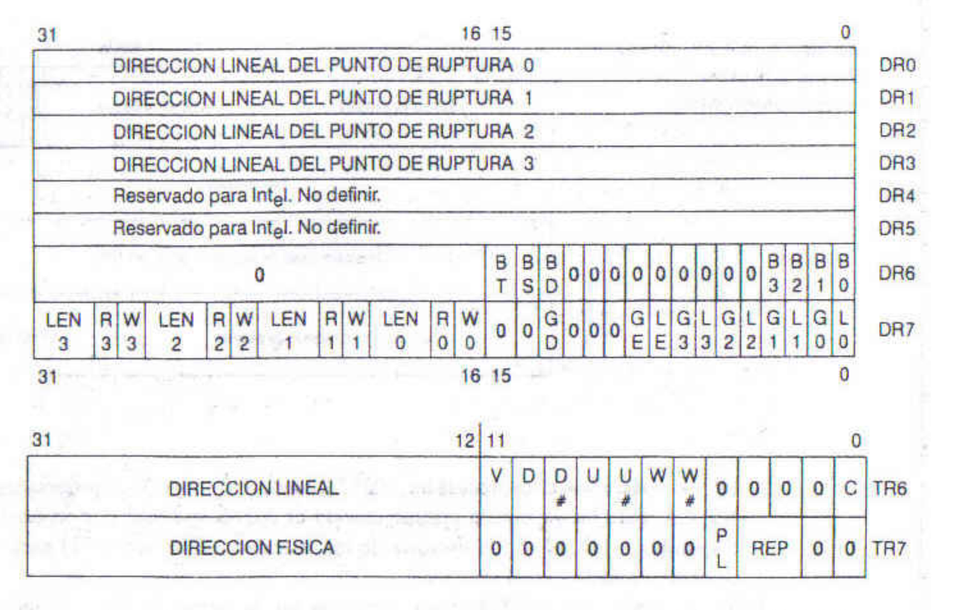


FIGURA 14–17 Los registros de depuración y de prueba del 80386. (Por cortesía de la Corporación Intel.)

1. BT — si está activa, la interrupción de depuración fue causada por una conmutación de tareas.
2. BS — si está activo (1), la interrupción de depuración fue causada por el bit de TF del registro de banderas.
3. BD — si está activo, la interrupción de depuración fue causada por un intento de lectura al registro de depuración con el bit GD activo. El bit de GD protege el acceso a los registros de depuración.
4. B3-B0 — indica cuál de las cuatro direcciones de los puntos de ruptura de depuración causó la interrupción de depuración.
5. LEN — cada uno de los cuatro campos de longitud pertenece a cada una de las cuatro direcciones de puntos de ruptura almacenadas en DRO-DR3. Estos bits definen aún más el tamaño del acceso a la dirección del punto de ruptura como 00 (byte), 01 (palabra), 11 (doble palabra).
6. RW — cada uno de los cuatro campos de lectura/escritura pertenece a cada una de las cuatro direcciones de los puntos de ruptura almacenadas en DR0-DR3. El campo RW selecciona la causa de la acción que habilitó una dirección de punto de ruptura como 00 (acceso por instrucción), 01 (escritura de datos), y 11 (lectura y escritura de datos).
7. GD — si está activo (1), GD evita cualquier lectura o escritura de un registro de depuración generando una interrupción de depuración. Este bit se desactiva (0) automáticamente durante la interrupción de depuración para que los registros de depuración puedan ser leídos o cambiados en caso de ser necesario.
8. GE — si está activo (1), selecciona una dirección de global para el punto de ruptura para cualquier dirección de los cuatro registros de punto de ruptura.

9. LE — si está activo (1), selecciona una dirección local para el punto de ruptura para cualquiera de los cuatro registros de dirección de punto de ruptura.

Los registros de prueba, TR6 y TR7, prueban el área de transformaciones frecuentes (TLB). El TLB se utiliza con la unidad de paginación dentro del 80386. El TLB contiene las transformaciones de las direcciones de la tabla de páginas más comúnmente utilizadas. El TLB reduce el número de lecturas de la memoria necesarias para buscar las entradas a la tabla de la transformación de páginas. El TLB contiene las 32 entradas de tabla de página más comúnmente utilizadas, y se prueba con los registros de prueba TR6 y TR7.

El registro de prueba TR6 contiene el campo del marcador (dirección lineal) de la TLB, y TR7 contiene la dirección física de la TLB. Para escribir una entrada a la TLB, realice los siguientes pasos:

1. Escriba en TR7 la dirección física deseada, y los valores de PL y REP.
2. Escriba en TR6 la dirección lineal, asegurándose que C = 0.

Para leer una entrada de TLB:

1. Escriba en TR6 la dirección lineal, asegurándose que C = 1.
2. Lea tanto TR6 y TR7. Si el bit PL indica una coincidencia, entonces los valores deseados de TR6 y TR7 indican los contenidos de la TLB.

Los bits encontrados en TR6 y TR7 indican las siguientes condiciones:

1. V — muestra que la entrada en la TLB es válida.
2. D — indica que la entrada en el TLB es no válida o corrupta.
3. U — un bit de usuario para la TLB.
4. W — indica que el área dirección por la entrada a la TLB se puede escribir.
5. C — selecciona una escritura (0) o búsqueda inmediata (1) para la TLB.
6. PL — indica una coincidencia si es un 1 lógico.
7. REP — selecciona qué bloque del RLB se escribe.

Vea la sección sobre la administración de memoria y la unidad de paginación para obtener más detalles sobre la función de la TLB.

14-3 EL CONJUNTO DE INSTRUCCIONES DEL 80386

El conjunto de instrucciones del 80386 contiene todas las instrucciones discutidas por el 8086 más un vasto número de instrucciones y modos de direccionamiento adicionales que aparecen a partir del 80386 y diferentes a los que fueron prestados en capítulos anteriores sobre el conjunto de instrucciones del microprocesador. Muchas de las nuevas variaciones se deben a los registros extendidos de 32 bits del 80386. Además de los registros de 8 y 16 bits encontrados en el 8086, también tenemos los siguientes registros extendidos de 32 bits: EAX, EBX, ECX, EDX, ESP,

EBP, EDI y ESI, junto con dos nuevos registros de segmento (selector): FS y GS. Estos registros adicionales incrementan el número de instrucciones válidas para el 80386.

Estos nuevos registros se utilizan con las instrucciones, exactamente como los registros de 8 y 16 bits se utilizan con el conjunto de instrucciones del 8086. Los registros de 32 bits se pueden direccionar en el modo de operación real o protegido. El Apéndice B, que describe el conjunto completo de instrucciones de todas las variaciones de la familia Intel, indica muchos ejemplos de las instrucciones del 80386. Además de los nuevos registros de 32 bits y de los nuevos registros de segmento, también hay 14 instrucciones nuevas (véase la tabla 14-1) y muchos modos de direccionamiento nuevos. Observe que todas estas instrucciones se explican en capítulos anteriores.

Notas sobre instrucciones más viejas

En el microprocesador 80286, la instrucción LMSW carga la palabra de estado de la máquina (MSW). En el 80386 MSW es CR0. Si planea modificar el contenido de CR0, no utilice la instrucción LMSW. Utilice la instrucción de MOV EAX,CR0 para transferir CR0 a EAX y luego cambiarlo con una instrucción AND u OR. Después de cambiar la imagen de CR0 dentro de EAX, la transfiere nuevamente a CR0 con la instrucción MOV CR0,EAX. Asimismo, la instrucción de SMSW nunca se debe de utilizar con los programas del 80386.

Por lo general, la mayoría de las instrucciones del 8086 se pueden ajustar para ser utilizadas en los registros de 32 bits del microprocesador 80386. Por ejemplo, si necesita cargar EAX con un 12H, utilice la instrucción de MOV EAX,12H. Vea el Apéndice B para una multitud de ejemplos sobre el uso de las instrucciones. Casi cualquier instrucción que utiliza un registro o apuntador de 8 o 16 bits puede utilizar un registro o apuntador de 32 bits.

Modos de direccionamiento nuevos

El número de modos de direccionamiento se ha incrementado sobre el número permitido por el microprocesador 8086. Los modos adicionales encontrados en el microprocesador 80386 inclu-

TABLA 14–1 Nuevas instrucciones del 80386

Instrucción	*Comentario*	*Ejemplo*
BSF	Rastreo de bit hacia delante	BSF EAX,DATA
BSR	Rastreo de bit hacia atrás	BSR EAX,LIST
BT	Prueba de bit	BT DATA,EAX
BTC	Prueba de bit y complemento	BTC AX,WATER
BTR	Prueba de bit y desactivación	BTR EBX,4
BTS	Prueba de bit e inactivación	BTS BX,2
LFS	Cargar FS	LFS DI,DATA
LGS	Cargar GS	LGS SI,FIELD
LSS	Cargar SS	LSS SP,STACK
MOVZX	Transferir con extensión de cero	MOVZX EAX,CX
MOVSX	Transferir con extensión de signo	MOVXS ECX,DL
SETcd	Activar el byte sobre condición	SETNC AL
SHLD	Corrimiento a la izquierda con doble precisión	SHLD AX,BX,8
SHRD	Corrimiento a la derecha con doble precisión	SHRD AX,BX,4

yen el uso de registros extendidos (32 bits) como direcciones de desplazamiento y apuntadores de 32 bits. Además, se han agregado algunas formas adicionales de indexación, llamadas *escalamiento*.

Los más obvios de los nuevos modos de direccionamiento utilizan los nuevos registros de segmento con el prefijo de cambio de segmento (FS: o GS:). La instrucción de MOV AL,FS:[DI] transfiere a AL los datos almacenados en el segmento FS en la dirección de desplazamiento DI. Cualquier instrucción que puede tener un prefijo puede utilizar FS o GS para direccionar con estos dos segmentos adicionales de la memoria.

Además de los modos de direccionamiento indirecto a la memoria utilizados con el microprocesador 8086, el 80386 contiene modos adicionales ilustrados en la tabla 14-2. Observe que bastantes modos adicionales de direccionamiento indirecto se agregan al conjunto de instrucciones del 80386. El 8086 permitía que la memoria se direccionara indirectamente sólo a través de DI, SI, BX o BP. Aún se permiten en el 80386, pero también se pueden utilizar cualquiera de los nuevos modos indicados en la tabla 14-2. Si estos nuevos modos de direccionamiento se utilizan en el modo real, sólo pueden direccionar el primer 1M byte de memoria.

Observe que la memoria puede direccionarse indirectamente utilizando un registro de 32 bits o utilizando directamente un desplazamiento de 32 bits. Esto incrementa en un factor enorme el número posible de instrucciones disponibles para el 80386.

TABLA 14-2 Modos de direccionamiento indirecto adicionales para el microprocesador 80386

Modo	*Segmento predeterminado*	*Ejemplo*
[EAX]	DS	ADD ECX,[EAX]
[EBX]	DS	SUB [EBX],AL
[ECX]	DS	MOV AX,[ECX]
[EDX]	DS	MOV AL,[EDX]
[EBP]	SS	MOV CL,[EBP]
[EDI]	DS	MOV [EDI],ECX
[ESI]	DS	MOV [ESI],EDI
d32	DS	MOV AL,DATA
[EAX + d8]	DS	ADD ECX,[EAX + 9]
[EBX + d8]	DS	SUB [EBX + 10H],AL
[ECX + d8]	DS	MOV AX,[ECX – 2]
[EDX + d8]	DS	MOV AL,[EDX – 12H]
[EBP + d8]	SS	MOV CL,[EBP + 1]
[EDI + d8]	DS	MOV [EDI – 33],ECX
[ESI + d8]	DS	MOV [ESI + 9],EDI
[EAX + d32]	DS	ADD ECX,TABLE[EAX]
[EBX + d32]	DS	SUB [EBX + 10000H],AL
[ECX + d32]	DS	MOV AX,ARRAY[ECX]
[EDX + d32]	DS	MOV AL,[EDX – 200000H]
[EBP + d32]	SS	MOV CL,[EBP +1]
[EDI + d32]	DS	MOV TABLE[EDI],ECX
[ESI + d32]	DS	MOV TABLE[ESI + 92],EDI

Notas: d8 = 8 bits de desplazamiento con signo y d32 = 32 bits de desplazamiento con signo.

En el microprocesador 8086, se pueden combinar sólo ciertos apuntadores para direccionar indirectamente la memoria: [BX + DI], [BX + SI], [BP + DI], y [BP + SI]. En el 80386 dos registros cualesquiera de 32 bits (excepto ESP) se pueden combinar para direccionar datos en la memoria. Estos modos adicionales de direccionamiento (véase la tabla 14-3) utilizan la siguiente forma [E?? + índice escalado], donde E?? es cualquier registro de 32 bits excepto ESP e índice escalado es cualquier registro escalado de 32 bits excepto ESP. La instrucción MOV EAX,[EBX + ECX] es un ejemplo de este nuevo modo de direccionamiento. Esta instrucción transfiere al registro EAX los datos de la memoria almacenados en el segmento de datos en la localidad direccionada por la suma de EBX y ECX.

El término índice de escalamiento significa que el segundo registro de 32 bits puede ser escalado (multiplicado) por un factor de 1X, 2X, 4X u 8X. Un factor de escalamiento 1X nunca se utiliza y es implícito como en la instrucción de MOV EAX,[EBX + ECX]. El factor de escalamiento multiplica el segundo registro por 1, 2, 4 u 8, para encontrar la localidad direccionada sin cambiar el valor del registro.

El factor de escalamiento se utiliza para direccionar elementos dentro de arreglos de datos. Supongamos que ECX accesa un arreglo llamado ARREGLO y que el número del elemento se localiza en EDX.

TABLA 14–3 Modo de direccionar indexado y escalado para el microprocesador 80386

Modo	*Segmento implícito*	*Ejemplo*
[EAX + scaled index]	DS	MOV BH,[EAX + 2*ECX]
[EBX + scaled index]	DS	ADD AL,[EBX + EAX]
[ECX + scaled index]	DS	INC BYTE PTR [ECX + 4*EAX]
[EDX + scaled index]	DS	MOV [EDX + 8*EAX],SP
[EBP + scaled index]	SS	MOV AL,[EBP + ECX]
[EDI + scaled index]	DS	MOV BL,[EDI + ESI]
[ESI + scaled index]	DS	SUB BYTE PTR [ESI + EBX],22H
[EAX + scaled index + d8]	DS	MOV BH,[EAX + 2*ECX + 3]
[EBX + scaled index + d8]	DS	ADD AL,[EBX + EAX – 2]
[ECX + scaled index + d8]	DS	INC BYTE PTR [ECX + 4*EAX – 9]
[EDX + scaled index + d8]	DS	MOV [EDX + 8*EAX – 10H],SP
[EBP + scaled index + d8]	SS	MOV AL,[EBP + ECX + 1AH]
[EDI + scaled index + d8]	DS	MOV BL,[EDI + ESI – 2]
[ESI + scaled index + d8]	DS	SUB BYTE PTR [ESI + EBX – 4],22H
[EAX + scaled index + d32]	DS	MOV BH,[EAX + 2*ECX + 200000H]
[EBX + scaled index + d32]	DS	ADD AL, TABLE[EBX + EAX]
[ECX + scaled index + d32]	DS	INC BYTE PTR ARRAY[ECX + 4*EAX]
[EDX + scaled index + d32]	DS	MOV ARRAY[EDX + 8*EAX + 100H],SP
[EBP + scaled index + d32]	SS	MOV AL,LIST[EBP + ECX]
[EDI + scaled index + d32]	DS	MOV BL,LIST[EDI + ESI + 10000H]
[ESI + scaled index + d32]	DS	SUB BYTE PTR ARRAY[ESI + EBX], 22H

Notas: d8 = 8 bits de desplazamiento con signo, d32 = 32 bits de desplazamiento con signo, e índice de escalamiento = EAX, EBX, ECX, EDX, EBP, EDI o ESI con un factor de escalamiento de 1X, 2X, 4X u 8X.

Si ARREGLO es de tamaño byte, entonces EDX direcciona el elemento del arreglo. Si ARREGLO es de tamaño palabra, EDX se tiene que multiplicar por un factor (escala) de 2 para direccionar la localidad correcta la memoria del elemento del arreglo. En un arreglo de tamaño palabra, el elemento de 0 se encuentra en la dirección con desplazamiento 0 y 1; el elemento 2 se encuentra en la dirección con desplazamiento 2 y 3; etcétera. Los factores de escalamiento de 4X y 8X se utilizan para indexar elementos en grupos de dobles y cuádruples palabras.

El ejemplo 14-2 indica un procedimiento corto que suma números de punto flotante con precisión sencilla en LIST1 y LIST2 y almacena el resultado en LIST3. Cada arreglo contiene 100H datos de doble palabra en este procedimiento de ejemplo que utiliza un direccionamiento escalado e indexado para accesar los elementos del arreglo. Si el factor de escalamiento se cambia de 4X a 8X, se suman datos de punto flotante de doble precisión. También observe que se resta un 4 dentro de cada modo de direccionamiento indirecto porque el último conteo en CX es un 1, no un cero. Observe que este programa está escrito asumiendo el modo de operación real en un sistema basado en DOS y no se hace ningún intento de colocar números dentro de los dos arreglos fuente LIST1 y LIST2.

EJEMPLO 14–2

```
                                  .386
                                  .387
                                  ;Procedimiento que suma grupos de datos de punto decimal flotante
0000                              CODE    SEGMENT USE16

                                          ASSUME  CS:CODE,DS:CODE

0000 0100 [                       LIST1   DD    100H DUP (?)
           00000000
                    ]
0400 0100 [                       LIST2   DD    100H DUP (?)
           00000000
                    ]
0800 0100 [                       LIST3   DD    100H DUP (?)
           00000000
                    ]
0C00                              ADDS    PROC FAR

0C00 1E                                   PUSH DS
0C01 8C C8                                MOV  AX,CS
0C03 8E D8                                MOV  DS,AX

0C05 66| 33 C0                            XOR  EAX,EAX            ;limpiar registros
0C08 66| 33 DB                            XOR  EBX,EBX
0C0B 66| 33 C9                            XOR  ECX,ECX
0C0E 66| 33 D2                            XOR  EDX,EDX

0C11 B8 0000 R                            MOV  AX,OFFSET LIST1    ;direccionar datos
0C14 BB 0400 R                            MOV  BX,OFFSET LIST2
0C17 BA 0800 R                            MOV  DX,OFFSET LIST3

0C1A B9 0100                              MOV  CX,100H            ;cargar contador

0C1D                              REPS:
0C1D 67& D9 44 88 FC                      FLD  DWORD PTR [EAX+4*ECX-4]
0C22 67& D8 44 8B FC                      FADD DWORD PTR [EBX+4*ECX-4]
```

```
0C27 67& D9 5C 8A FC          FSTP DWORD PTR [EDX+4*ECS-4]
0C2C E2 EF                    LOOP REPS

0C2E 1F                       POP  DS
0C2F CB                       RET

0C30                ADDS      ENDP

0C30                CODE      ENDS

                              END  ADDS
```

Interrupciones

Aunque este tema contiene información para interfaces, también contiene instrucciones que se presentaron con el conjunto de instrucciones. La figura 14-18 muestra las interrupciones predefinidas para el 80386. Estas deben ser comparadas a las interrupciones del 8086 en el Capítulo 10. El 80386 contiene el tipo de interruptor número 14 y 16 que no están definidos para las versiones anteriores del microprocesador. El tipo 14 ocurre para una falla del mecanismo de paginado y el tipo 16 para un error del coprocesador.

Frecuentemente le llamamos a las interrupciones generadas en las entradas INTR y NMI *interrupciones duras (hardware)* y de cualquier otro tipo se llaman *excepción*. Las excepciones son generadas por eventos internos, mientras que las interrupciones son generadas por eventos externos. Una **excepción** es una excepción al progreso normal de un programa. Las interrupciones y excepciones predefinidas arriba del nivel 5 se definen en la siguiente lista:

1. Tipo 6 — cualquier instrucción ilegal causa una interrupción de tipo 6.
2. Tipo 7 — si está activo el bit EM en el CR0, la interrupción de tipo 7 ocurre para cualquier instrucción del coprocesador aritmético. Si está activo el bit MP de CR0 y está activo TS por una conmutación de tareas, la instrucción WAIT también causará una interrupción de tipo 7.
3. Tipo 8 — ocurre una falta doble o excepción de aborto cada vez que el 80386 detecta una interrupción de tipo 10, 11, 12 o 13 al mismo tiempo que ocurre un interrupción distinta a la 14.
4. Tipo 9 — esta excepción ocurre si la dirección de un operando para el coprocesador aritmético se "enrolla" de la localidad FFFFH a la 0000H o de la 0000H a la FFFFH en el modo real. En el modo protegido, esta excepción ocurre si la dirección se enrolla de la localidad FFFFFFFFH a la 00000000H o de la 00000000H a la FFFFFFFFH.
5. Tipo 10 — si se accesa un TSS no válido, ocurre la excepción de tipo 10. Ocurre un TSS no válido cada vez que el selector está fuera del límite de la tabla, si un código en el segmento de pila está fuera del límite de la tabla, si no se puede escribir a una pila, o si el nivel de privilegio solicitado no es igual al nivel de privilegio actual del TSS accesado.
6. Tipo 11 — ocurre si el bit de P del descriptor indica que el segmento no está presente (P = 0).
7. Tipo 12 — ocurre si el descriptor SS accesa un segmento que no está presente o si cualquier operación de pila causa una violación del límite.
8. Tipo 13 — ocurre por cualquiera de las siguientes razones: exceder el límite del segmento, escribir a un segmento de datos o de código de sólo lectura, cargar un selector con

Función	Número de interrupción	Instrucción que puede causar excepción	La dirección de regreso apunta a la instrucción que falló	Tipo
Error de división	0	DIV, IDIV	SI	FALLA
Excepción de depuración	1	Cualquier instrucción	SI	TRAMPA*
Interrupción NMI	2	INT 2 o NMI	NO	NMI
Interrupción de un byte	3	INT	NO	TRAMPA
Interrupción por sobreflujo	4	INTO	NO	TRAMPA
Verificación de límites de arreglo	5	BOUND	SI	FALLA
Código de funcionamiento no válido	6	Cualquier instrucción ilegal	SI	FALLA
Dispositivo no disponible	7	ESC, WAIT	SI	FALLA
Falta doble	8	Cualquier instrucción que puede generar una excepción		ABORTO
Sobreescritura en el segmento del coprocesador	9	ESC	NO	ABORTO
TSS no válido	10	JMP, CALL, IRET, INT	SI	FALLA
Segmento no presente	11	Instrucciones para el registro de segmento	SI	FALLA
Falla de pila	12	Referencias de pila	SI	FALLA
Falla de protección general	13	Cualquier referencia de la memoria	SI	FALLA
Falla de página	14	Cualquier acceso de la memoria o recuperación de código	SI	FALLA
Error del coprocesador	16	ESC, WAIT	SI	FALLA
Reservado para Intel	17-32			
Interrupción de dos bytes	0-255	INT n	NO	TRAMPA

* Algunas excepciones de depuración pueden reportar tanto trampas en la instrucción anterior, y fallas en la siguiente instrucción.

FIGURA 14-18 Los vectores de interrupción predefinidos del microprocesador 80386. (Por cortesía de la Corporación Intel.)

un descriptor del sistema, leer un segmento de código sólo de ejecución, cambiar a una tarea ocupada, violar el nivel de privilegio de un segmento de datos, o cargar CR0 con PG = 1 y PE = 0.

9. Tipo 14 — una excepción de la falla de página ocurre si PG = 1 y si el nivel de privilegio es el incorrecto o si la tabla de páginas o directorio contienen un cero.
10. Tipo 16 — la terminal de $\overline{\text{ERROR}}$ del 80386 causa una excepción de tipo 16.

Las interrupciones duras del 80386 son iguales que para el 8086. Para mejorar su comprensión de las entradas INTR y NMI, vea el Capítulo 10 para una discusión completa de estas entradas de interrupción.

14–4 ADMINISTRACION DE MEMORIA DEL 80386

La unidad de administración de la memoria (MMU) dentro del 80386 es similar a la del 80286, excepto que el 80386 tiene una unidad de paginación que no se encuentra en el 80286. La MMU realizará la tarea de convertir direcciones lógicas conforme aparecen como las salidas de un programa, a direcciones físicas que accesan una localidad de la memoria física localizada en cualquier parte dentro del sistema de memoria. El 80386 puede utilizar un mecanismo de paginación para asignar cualquier dirección física a cualquier dirección lógica. Lo que esto significa es que aunque el programa está accesando la localidad A0000H de la memoria con una instrucción, la dirección física real puede ser la localidad 100000H de la memoria, o cualquier otra localidad si está habilitada la paginación. La paginación permite virtualmente que cualquier programa, escrito para operar en cualquier localidad de la memoria, funcione en un 80386 porque cualquier localidad física se puede convertir en cualquier localidad lógica. El 80286 no tenía esta flexibilidad. La paginación se utiliza en versiones del DOS más nuevas para relocalizar la memoria del 80386 y del 80486 a direcciones arriba de FFFFFH a espacios entre la ROM y las localidades D0000-DFFFFH y en otras áreas conforme estén disponibles.

Descriptores y selectores

Antes de tratar la unidad de paginación de la memoria, el descriptor y el selector se analizan para el microprocesador 80386. El 80386 utiliza descriptores de la misma forma que el 80286 (véase el Capítulo 1). En ambos microprocesadores, un **descriptor** es una serie de 8 bytes que describe y localiza un segmento de la memoria. Un **selector** (registro de segmento) se utiliza para indexar un descriptor en una tabla de descriptores. La diferencia principal entre el 80286 y el 80386 es que hay dos selectores adicionales (FS y GS). Los descriptores del 80386 utilizan también una dirección base de 32 bits y un límite de 20 bits en lugar de una dirección base de 24 bits y un límite de 16 bits, como se encuentra en el 80286.

El 80286 direcciona un espacio de la memoria de 16M bytes con su dirección base de 24 bits y tiene un límite de longitud del segmento de 64K bytes debido al límite de 16 bits. El 80386 direcciona un espacio de memoria de 4G bytes con su dirección base de 32 bits y tiene un límite de longitud del segmento de 1M bytes o 4G bytes, debido al límite de 20 bits que se utiliza en dos formas diferentes. El límite de 20 bits puede direccionar un segmento con una longitud de 1M byte si el bit de granularidad (G) = 0. Cuando G = 1, el límite de 20 bits permite una longitud de segmento de 4G bytes.

El bit de granularidad se encuentra en el descriptor del 80386. Cuando G = 0, el número almacenado en el límite se interpreta directamente como límite, permitiéndole contener cualquier límite entre 00000H y FFFFFH para un tamaño de segmento de hasta 1M bytes. Cuando G = 1, el número almacenado en el límite se interpreta como 00000XXXH a FFFFFXXXH, donde XXX es 000H. Esto permite que el límite del segmento varíe de 0 bytes a 4G bytes en etapas de 4K bytes. Un límite de 00001H indicará que el límite es de 4K bytes cuando G = 1, y 1 byte cuando G = 0.

La figura 14-19 muestra la forma en que el 80386 direcciona un segmento de la memoria en el modo protegido utilizando un selector y un descriptor. Observe que esto es idéntico a la forma en que un segmento es direccionado por el 80286. La diferencia es el tamaño del segmento accesado por el 80386. El selector utiliza sus 13 bits de más a la izquierda para direccionar un descriptor en una tabla de descriptores. El bit TI indica una tabla de descriptores locales (TI = 1)

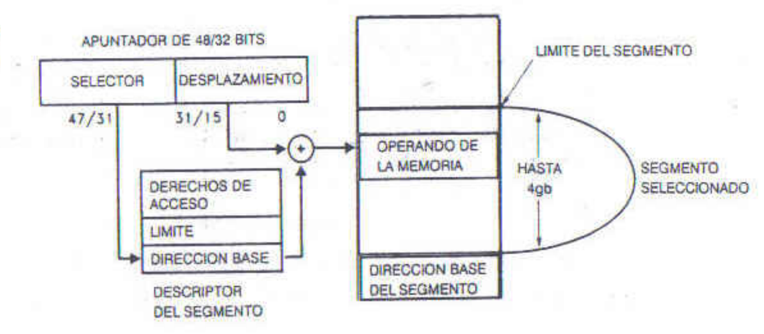


FIGURA 14-19 Direccionamiento en modo protegido utilizando un registro de segmento como selector. (Por cortesía de la Corporación Intel.)

o globales (TI = 0). Los 2 bits de más a la derecha del selector definen el nivel de privilegio solicitado por el acceso.

Debido a que el selector utiliza un código de 13 bits para direccionar un descriptor, se encuentran a lo más 8,192 descriptores en cada tabla, local o global. Debido a que cada segmento, en un 80386, puede tener 4G bytes de longitud, se pueden direccionar 16,384 segmentos a la vez con dos tablas de descriptores. Esto permite al 80386 direccionar un tamaño de memoria virtual de 64T bytes. Por supuesto, sólo 4G bytes de la memoria realmente existen en el segmento de la memoria (1T byte = 1,024G bytes). Si un programa requiere de más de 4G bytes de memoria a la vez, se puede intercambiar entre el sistema de la memoria y una unidad de disco u otra forma de almacenamiento de grandes volúmenes. El intercambio con los discos es rutinario en los sistemas operativos como Windows o el OS/2.

Al igual que con el 80286, el 80386 utiliza las tablas tanto para los descriptores globales (GDT) como para los locales (LDT). Ambas máquinas también utilizan una tercera tabla para interrupciones (IDT), descriptores o compuertas. Los descriptores para el 80386 son diferentes del 80286, la comparación de estos descriptores se ilustra en la figura 14-20. Observe cómo ambos descriptores utilizan los mismos datos para los primeros 6 bytes. Esto permite que la programación del 80286 sea compatible hacia arriba con el 80386. (Recuerde que un descriptor 80286 utilizó 00H para sus dos bytes más significativos.) La dirección base es de 32 bits en el 80386, el límite es de 20 bits, y el bit G selecciona el multiplicador del límite (1 vez o 4K veces). Los campos del descriptor para el 80386 se definen como sigue:

1. Base (B0-B31) — define la dirección inicial del segmento de 32 bits dentro del espacio de direccionamiento físico del microprocesador 80386.
2. Límite (L0-L19) — define el límite del segmento en unidades de bytes si el bit G = 0, o en unidades de 4K bytes si G = 1. Esto permite que un segmento pueda ser de cualquier longitud desde 1 byte a 1M bytes si G = 0 y desde 4K bytes a 4G bytes en etapas de 4K bytes si G = 1. Recuerde que el límite indica el último byte de un segmento.
3. Derecho de acceso — determina el nivel de privilegio, y otra información sobre el segmento. Este byte varía con diferentes tipos de descriptores y está elaborado con cada tipo de descriptor.
4. G — el bit de granularidad selecciona un multiplicador de 1 o 4K veces para el límite del campo. Cuando G = 0, el multiplicador es 1 y si G = 1, el multiplicador es 4K.
5. D — selecciona el tamaño de registro implícito. Si D = 0, los registros son de 16 bits de ancho como en el 80286 y si D = 1, son de 32 bits de ancho como en el 80386. Este bit

Descriptor 80286

Reservado		6
Derechos de acceso	Base (B23–B16)	4
Base (B15–B0)		2
Límite (L15-L0)		0

Descriptor 80386

Base (B24–B31)	G	D	O	AVL	Límite (L16-L19)	6
Derechos de acceso	Base (B23–B16)					4
Base (B15–B0)						2
Límite (L15-L0)						0

FIGURA 14–20 Los descriptores para los microprocesadores 80286 y 80386.

determina si son necesarios los prefijos para datos de 32 bits y registros índice. Cuando D = 0, entonces se requiere de un prefijo para direccionar los registros de 32 bits y para utilizar los apuntadores de 32 bits. Si D = 1, entonces se necesita un prefijo para accesar los registros de 16 bits y los apuntadores de 16 bits. Los directivos USE16 y USE32 junto con la declaración SEGMENT en lenguaje ensamblador, controlan la inicialización del bit D. En el modo real, siempre se supone que los registros son de 16 bits de ancho, así que cualquier instrucción que solicita un registro o apuntador de 32 bits debe tener un prefijo. Observe que las versiones actuales del DOS, Windows[1], y OS/2[2] suponen que D = 0.

6. AVL — este bit está disponible para el sistema operativo para ser utilizado de cualquier manera que se considere pertinente. Frecuentemente indica que el segmento descrito por el descriptor está disponible.

En el microprocesador 80386 aparecen descriptores de dos formas: el descriptor de segmento y el descriptor de sistema. El descriptor de segmento define los segmentos de datos, pila y código y el descriptor de sistema define información sobre las tablas, tareas y compuertas del sistema.

Descriptores de segmento. La figura 14-21 muestra el descriptor de segmento. Este descriptor tiene la forma general indicada en la figura 14-20, pero los bits de derechos de acceso están definidos para indicar cómo el segmento de datos, pila o código están descritos por las funciones del descriptor. La posición de bit 4 del byte de derechos de acceso determina si el descriptor es un descriptor de segmento de código o datos (S = 1) o un descriptor de segmento de sistema (S = 0).

Sigue una descripción de los bits de derechos de acceso y su función en el descriptor de segmento:

1. P — Presente: es un 1 lógico para indicar que el segmento está presente. Si P = 0 y el segmento es accesado a través del descriptor, ocurre una interrupción de tipo 11. Esta interrupción indica que se accesó un segmento que no está presente en el sistema.
2. DPL — Descriptor del nivel de privilegio: instala el nivel de privilegio del descriptor donde 00 tiene el privilegio más alto y 11 tiene el más bajo. Esto se utiliza para proteger el acceso

---

[1]Windows es una marca registrada de Microsoft Corporation.
[2]OS/2 es una marca registrada de International Business Machines Incorporated.

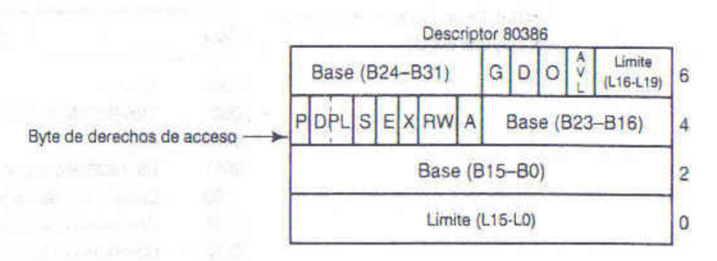


FIGURA 14-21 Formato del descriptor de segmento del 80386.

a los segmentos. Si un segmento se accesa con un nivel de privilegio que es más bajo (más alto en número) que el DPL, ocurre una interrupción de violación de privilegio. Los niveles de privilegio se utilizan en sistemas multiusuario para evitar acceso a una área de la memoria del sistema.

3. S — Segmento: indica un descriptor de segmento de datos o código (S = 1) o un descriptor de segmento de sistema (S = 0).
4. E — Ejecutable: selecciona un segmento de datos (pila) (E = 0) o un segmento de código (E = 1). E también define la función de los dos bits siguientes (X y RW).
5. X — si E = 0 entonces X indica la dirección de expansión para el segmento de datos. Si X = 0 el segmento se expande hacia arriba como un segmento de datos y si X = 1, el segmento se expande hacia abajo como un segmento de pila. Si E = 1, entonces X indica si el nivel de privilegio del segmento de código es ignorado (X = 0) u observado (X = 1).
6. RW — Lectura/Escritura: si E = 0 entonces RW indica que se puede escribir en el segmento de datos (RW = 1) o no se puede escribir (RW = 0). Si E = 0 entonces RW indica que el segmento de código se puede leer (RW = 1) o no leer (RW = 0).
7. A — Accesado: este bit se activa cada vez que el microprocesador accesa el segmento. A veces lo utiliza el sistema operativo para mantener un registro de cuáles segmentos se han accesado.

Descriptor de sistema. El descriptor de sistema se muestra en la figura 14-22. Hay 16 tipos de descriptores posibles para el sistema (véase la tabla 14-4 para los distintos tipos de descriptores), pero no todos se utilizan con el microprocesador 80386. Algunos de estos tipos se definen para el 80286, así que la programación del 80286 es compatible con el 80386. Algunos de los tipos son nuevos y originales del 80386. Algunos todavia tienen que ser definidos y están reservados para futuros productos de Intel.

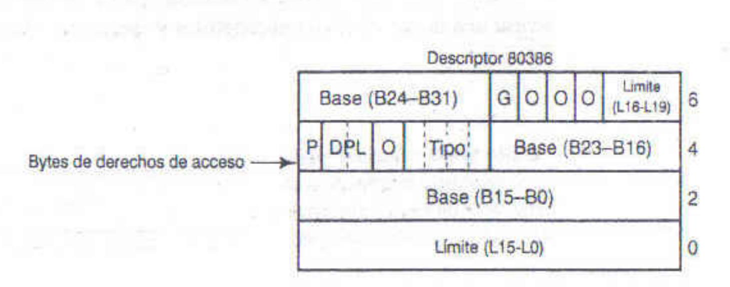


FIGURA 14-22 El formato general del descriptor de sistema 80386.

ble del caché del registro de segmento. Mientras que el selector permanezca igual en el registro de segmento, no se necesitan accesos adicionales a la tabla de descriptores. La operación de recuperación de un descriptor nuevo de la tabla de descriptores es invisible al programa, porque el microprocesador realiza esto automáticamente cada vez que se cambia el contenido del registro de segmento en el modo protegido.

La figura 14-24 es una muestra de una tabla de descriptores globales (GDT), que está almacenada en la dirección de memoria 00010000H, se accesa a través del registro de segmento y su selector. Esta tabla contiene cuatro entradas. La primera es un descriptor nulo (0). El descriptor 0 siempre debe ser un descriptor nulo. Las otras entradas direccionan varios segmentos en el sistema de memoria del modo protegido 80386. En esta ilustración, el registro del segmento de datos contiene 0008H. Esto significa que el selector está indexando la localidad 1 de los descriptores de la tabla global de descriptores (TI = 0), con un nivel de privilegio de solicitud de 00. El descriptor 1 está localizado 8 bytes arriba de la dirección base de la tabla de descriptores en la localidad 00010008H. El descriptor localizado en esta localidad de la memoria accesa una dirección base 00200000H y un límite de 100H. Esto significa que este descriptor direcciona las localidades de memoria 00200000H-00200100H. Debido a que éste es el registro DS (segmento de datos), esto significa que el segmento de datos se localiza en estas localidades del sistema de memoria. Si los datos se accesan fuera de estos límites, ocurre una interrupción.

EAX		AH	AL	AX	
EBX		BH	BL	BX	
ECX		CH	CL	CX	
EDX		DH	DL	DX	Registros de propósito general
ESP		SP			
EBP		BP			
EDI		DI			
ESI		SI			
		CS			
		DS			
		ES			
		SS			Registros de segmento (selector)
		FS			
		ES			
EIP		IP			
EFLAGS		FLAGS			Registros de control

FIGURA 14-24 El segmento de datos utilizado para contener un selector que direcciona la entrada 1 de los descriptores globales.

TABLA 14-4 Tipos de descriptor del sistema 80386

Tipo	*Descripción*
0000	No válido
0001	TSS 80286 disponible
0010	LDT
0011	TSS 80286 ocupado
0100	Compuerta de llamada del 80286
0101	Compuerta de tarea (80286 o 80386)
0110	Compuerta de interruptor del 80286
0111	Compuerta de trampa del 80286
1000	No válido
1001	TSS 80386 disponible
1010	Reservado para productos futuros de Intel
1011	TSS 80386 ocupado
1100	Compuerta de solicitud del 80386
1101	Reservado para productos futuros de Intel
1110	Compuerta de interrupción del 80386
1111	Compuerta de trampa del 80386

Tablas de descriptores

Las tablas de descriptores definen todos los segmentos utilizados en el 80386 cuando se operan en el modo protegido. Hay tres tipos de tablas de descriptores: la tabla descriptores globales (GDT), la tabla de descriptores locales (LDT), y la tabla de descriptores de interrupciones (IDT). Los registros utilizados por el 80386 para direccionar estas tres tablas se llaman registro de tabla de descriptores globales (GDTR), registro de tabla de descriptores locales (LDTR), y registro de tabla descriptores de interrupciones (IDTR). Estos registros se cargan respectivamente con las instrucciones LGDT, LLDT y LIDT.

La **tabla de descriptores** es un arreglo de datos de longitud variable con cada entrada conteniendo un descriptor de 8 bytes de longitud. Las tablas de descriptores locales y globales contienen 8,192 entradas cada una y la tabla de descriptores interrupciones contiene 256 entradas. Un descriptor se indexa desde la tabla local o global del descriptor por el selector que aparece en un registro de segmento. La figura 14-23 muestra un registro de segmento y el selector que tiene en el modo protegido. Los 13 bits de más a la izquierda indexan un descriptor, el bit TI selecciona la tabla de descriptores locales (TI = 1) o globales (TI = 0), y los bits RPL indican el nivel de privilegio solicitado.

Cada vez que se coloca un selector nuevo en uno de los registros de segmento, el 80386 accesa una de las tablas de descriptores y carga automáticamente el descriptor en la parte invisi-

FIGURA 14-23 Cualquier registro de segmento mostrando el selector, el bit de TI y los bits del nivel de privilegio (RPL) solicitado.

15 ... 3	2	1 0
Selector	TI	RPL

Registro de segmento

La tabla de descriptores locales (LDT) se accesa de la misma manera que la tabla de descriptores globales (GDT). La única diferencia en el acceso es que el bit de TI se desactiva para un acceso global y se activa para un acceso local. Hay otras diferencias si se examinan los registros de las tablas de descriptores locales y globales. El registro de la tabla de descriptores globales (GDTR) contiene la dirección base de la tabla de descriptores globales y el límite. El registro de la tabla de descriptores locales (LDTR) sólo contiene un selector y es de 16 bits de ancho. El contenido del LDTR direcciona un descriptor del sistema de tipo 0010 que contiene la dirección base y el límite de la LDT. Este esquema permite una tabla global para todas las tareas, pero también muchas tablas locales, una o más para cada tarea, si es necesario. Los descriptores globales describen la memoria para el sistema, mientras que los descriptores locales describen la memoria para las aplicaciones o tareas.

La tabla de descriptores de interrupciones (IDT) se direcciona como la GDT almacenando la dirección base y el límite en el registro de la tabla de descriptores de interrupciones (IDTR). La diferencia principal entre la GDT y la IDT es que la IDT contiene sólo compuertas para interrupciones en lugar de descriptores de segmento y del sistema, como la GDT y la LDT.

La figura 14-25 muestra el descriptor de una compuerta, una forma especial del descriptor del sistema descrito anteriormente. (Vea la tabla 14-4 para los diferentes tipos de descriptores de compuerta.) Observe que el descriptor de compuerta contiene una dirección de desplazamiento de 32 bits, un contador de palabras y un selector. La dirección de desplazamiento de 32 bits apunta a la localidad del procedimiento de servicio para la interrupción o a otro procedimiento. El contador de palabras indica cuántas palabras se transfieren de la pila del procedimiento que llamó, a la pila del procedimiento que se accesa por medio de una compuerta de llamada. El campo del contador de palabras no se utiliza con una compuerta de interrupción. El selector se utiliza para indicar la localidad del segmento de estado de la tarea (TSS) en la GDT o la LDT si es un procedimiento local.

Cuando se accesa una compuerta, el contenido del selector se carga dentro del registro de tareas (TR). La aceptación de la compuerta depende de los niveles de privilegio y de prioridad. Una instrucción de retorno (RET) termina un procedimiento de llamada a una compuerta y el regreso de una instrucción de regreso de interrupción (IRET) terminará un procedimiento de llamada a una compuerta de interrupción. Las tareas normalmente se accesan con una instrucción CALL o INT.

La diferencia entre las interrupciones de modo real y las de modo protegido es que la tabla de vector de interrupción es una IDT en el modo protegido. La IDT contiene hasta 256 niveles de interrupción, pero cada nivel se accesa a través de una compuerta de interrupción en lugar de un vector de interrupción. Por lo tanto, la interrupción de tiempo 2 se localiza en el descriptor 2 de la

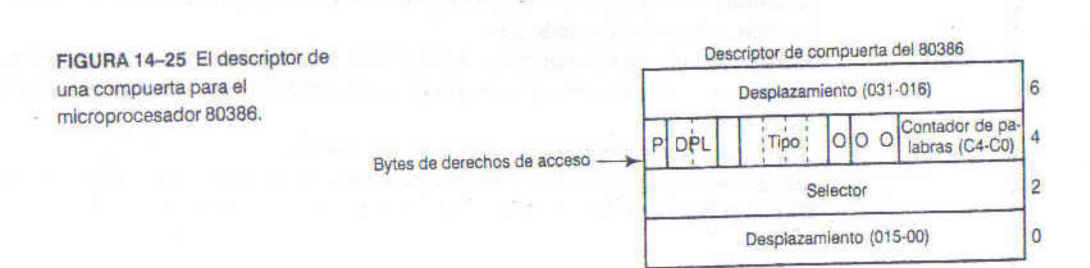


FIGURA 14-25 El descriptor de una compuerta para el microprocesador 80386.

IDT 16 localidades arriba de la dirección base de la IDT. Esto también significa que el primer 1K byte de la memoria ya no tiene que tener los vectores de interrupción como en el modo real ya que la IDT se puede localizar en cualquier localidad del sistema de la memoria.

El segmento de estado de tarea (TSS)

El descriptor del segmento de estado de tarea (TSS) contiene información sobre la localidad, tamaño y nivel de privilegio del segmento de estado de tarea, como cualquier otro descriptor. La diferencia es que el TSS descrito por el descriptor TSS no contiene ni datos ni códigos. Contiene el estado de la tarea y el eslabonamiento para que las tareas puedan anidarse, o sea, que una tarea puede llamar una segunda que a su vez puede llamar una tercera, etcétera. El descriptor del TSS lo direcciona el registro de tareas (TR). El contenido del TR se cambia por medio de la instrucción LTR o cada vez que el programa de modo protegido ejecuta una instrucción JMP o de CALL. La instrucción LTR se utiliza para accesar inicialmente una tarea durante la inicialización del sistema. Después de la inicialización, las instrucciones CALL o JUMP normalmente ejecutan una conmutación de tareas. En la mayoría de los casos se utiliza la instrucción CALL para iniciar una nueva tarea.

El TSS se muestra en la figura 14-26. Como se puede ver, el TSS es una sección formidable de la memoria que contiene muchos diferentes tipos de información. La primera palabra del TSS está etiquetado de retorno. Este es el selector que se utiliza en un retorno (RET o IRET) para enlazar nuevamente al TSS anterior, cargando el selector del lazo de retorno en el TR. La siguiente palabra tiene que contener un 0. De la segunda a la séptima dobles palabras contienen los valores de ESP y ESS para los niveles de privilegio de 0-2. Estos son necesarios en caso de que se interrumpa la tarea actual para que se puedan accesar las pilas de estos niveles de privilegio (PL). La octava palabra (desplazamiento 1CH) contiene el contenido de CR3 que almacena la dirección base del registro del directorio de página del estado anterior. Esta se tiene que restablecer si está en efecto la paginación. El contenido de las siguientes 17 dobles palabras se carga dentro de los registros indicados. Cada vez que se accesa una tarea, todo el estado de la máquina se almacena en estas localidades de la memoria y luego se vuelve a cargar desde las mismas localidades en el nuevo TSS. La última palabra (desplazamiento 66H) contiene la dirección base del mapa de bits del permiso para entrada/salida.

El mapa de bits de permiso de entrada/salida permite al TSS bloquear las operaciones de entrada/salida para inhibir las direcciones de puertos de entrada/salida por medio de la interrupción de permiso denegado de entrada/salida. La interrupción para el permiso denegado es de tipo 13, la interrupción de protección general a fallas. La dirección base para el mapa de bits de permiso de entrada/salida, es la dirección del desplazamiento inicial del TSS. Esto permite que muchos TSS, utilicen el mismo mapa de permiso.

Cada mapa de bits de permiso de entrada/salida es de 64K bits (8K bytes) de longitud, y empieza en la dirección del desplazamiento indicada por la dirección base del mapa de bits de permisos de entrada/salida. El primer byte del mapa de bits del permiso de entrada/salida contiene el permiso de entrada/salida para los puertos de entrada/salida 0000H-0007H. Los bits de más a la derecha contienen el permiso para el número de puerto 0000H y los de más a la izquierda para el número de puerto 0007H. Esta secuencia continúa para la última dirección de puerto (FFFFH) almacenada en el bit de más a la izquierda del último byte del mapa de bits de permisos de entrada/salida. Un 0 lógico colocado en un mapa de bits del permiso de entrada/salida activa las direcciones de puertos de entrada/salida, mientras que un 1 lógico inhibe o bloquea las direcciones de puertos de entrada/salida.

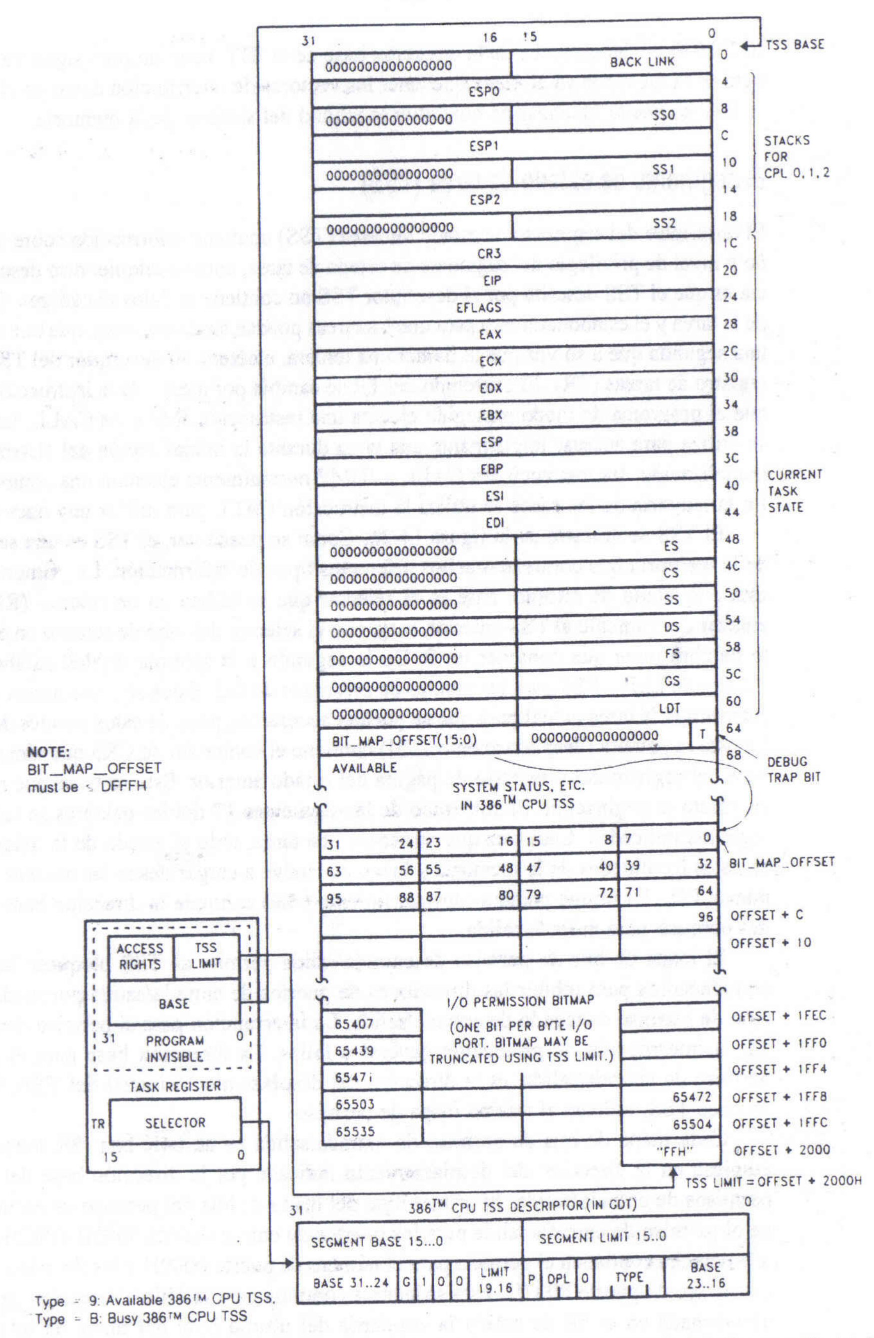


FIGURA 14–26 El descriptor para el segmento del estado de tareas (TSS). (Por cortesía de la Corporación Intel.)

Como repaso de la operación de una conmutación de tareas, que requiere de sólo 17 μs para ejecutarse, listamos los siguientes pasos:

1. La compuerta contiene la dirección del procedimiento o localidad a la que se brincó por la conmutación de tareas. También contiene el número de selector del descriptor del TSS y el número de palabras transferidas de la pila del procedimiento que llamó al área de pila del usuario para el paso de parámetro.
2. El selector se carga al TR desde la puerta. (Este paso se realiza por medio de CALL o JMP que se refiere a un descriptor de TSS válido.)
3. El TR selecciona el TSS.
4. El estado actual se almacena en el TSS actual y el nuevo TSS se accesó con el estado de la nueva tarea (todos los registros) cargados en el microprocesador. El estado actual se almacena en el selector de TSS que se encuentra actualmente en el TR. Una vez que el estado actual se almacena, se carga un nuevo valor (por medio de JMP o CALL) para el selector de TSS en TR y el nuevo estado se carga del nuevo TSS.

Para regresar de una tarea se realizan los siguientes pasos:

1. El estado actual del microprocesador se almacena en el TSS actual.
2. El selector de línea retroactiva se carga al TR para direccionar el TSS anterior, para que se pueda regresar al estado anterior de la máquina y se pueda restablecer, regresar y restaurar en el microprocesador. El regreso de una llamada al TSS se realiza por medio de la instrucción IRET.

14-5 CAMBIANDO AL MODO PROTEGIDO

Para poder cambiar la operación del 80386 del modo real al modo protegido, se deben seguir varios pasos. La operación en modo real se accesa después de una reinicialización dura o cambiando el bit PE a un 0 lógico en CR0. El modo protegido se accesa colocando un 1 lógico dentro del bit de PE del CR0, pero antes de que se realice esto se tienen que inicializar algunas otras cosas. Los siguientes pasos realizan el cambio del modo real al modo protegido:

1. Inicialice la tabla de descriptores de interrupciones para que contenga las compuertas válidas para interrupciones para por lo menos los primeros 32 tipos de interrupción. La IDT puede tener hasta 256 compuertas de interrupción de ocho bytes definiendo los 256 tipos de interrupción, y frecuentemente lo hace.
2. Inicialice la tabla de descriptores globales (GDT) para que contenga un descriptor nulo en el descriptor 0, y descriptores válidos para un segmento de código, uno de pila y uno de datos, por lo menos.
3. Cambie al modo protegido activando el bit PE de CR0.
4. Realice un intrasegmento JMP (cercano) para borrar la cola interna de instrucciones y cargar TR con el descriptor base TSS.

5. Cargar todos los selectores de datos (registros de segmento) con sus valores iniciales para los selectores.
6. El 80386 está ahora operando en el modo protegido utilizando los descriptores de segmento que se definen en GDT e IDT.

La figura 14-27 muestra el mapa para la memoria del sistema en modo protegido inicializado utilizando los pasos 1-5. El listado del programa para esta tarea se da en el ejemplo 14-3. Este sistema contiene un descriptor del segmento de datos y un descriptor de segmento de código con cada segmento inicializado a una longitud de 4G bytes de largo.

EJEMPLO 14–3

```
                    .386p
                    ;Este programa hace que el 80386 entre al modo protegido.
                    ;No muestra los descriptores de las interrupciones o cualquier
                    ;programa que será ejecutado en el modo protegido.

0000                DATA    SEGMENT AT 0000H   ;dirección del segmento = 0000H

0000                        ORG   0000H

                    ;los primeros 32 vectores de interrupción colocados aquí

                            ORG   0100H

                    ;tabla de descriptores globales
```

Dirección	Contenido
FFFFFFFF – FFFFFFF0	Programa de reinicialización
	Segmento de datos y código
	Descriptores globales
00000100 – 0000000	Descriptores de las interrupciones

FIGURA 14–27 El mapa de la memoria para el ejemplo 14-3.

```
0100 0000000000000000       DES0    DQ   0                    ;descriptor nulo

                            ;Descriptor del segmento de código

0108 FFFF                   DES1    DW   0FFFFH               ;límite 4 G
010A 0000                           DW   0                    ;dirección base = 00000000H
010C 00                             DB   0
010D 9E                             DB   9EH                  ;segmento de código
010E 8F                             DB   8FH                  ;G = 1
010F 00                             DB   0

                            ;Descriptor del segmento de datos

0110 FFFF                   DES2    DW   0FFFFH               ;límite 4 G
0112 0000                           DW   0                    ;dirección base = 00000000H
0114 00                             DB   0
0115 92                             DB   92H                  ;segmento de datos
0116 8F                             DB   8FH                  ;G = 1
0117 00                             DB   0

                            ;Datos de la tabla IDT

0118 00FF                   IDT     DW   0FFH                 ;inicializar límite a FFH
011A 00000000                       DD   0                    ;dirección base 0H

                            ;Datos de la tabla GDT

011E 0017                   GDT     DW   17H                  ;inicializar límite a 17H
0120 00000100                       DD   100H                 ;dirección base 100H

0124                        DATA    ENDS

0000                        CODE    SEGMENT USE16

                                    ASSUME CS:CODE,DS:DATA

0000 B8 ---- R              START:  MOV  AX,DATA              ;cargar DS
0003 8E D8                          MOV  DS,AX
0005 67| 0F 01 1D 00000118 R            LIDT FWORD PTR IDT ;load IDTR
000D 67| 0F 01 15 0000011E R            LGDT FWORD PTR GDT ;load GDTR

0015 0F 20 C0                           MOV  EAX,CR0          ;inicializar PE
0018 0C 01                              OR   AL,1
001A 0F 22 C0                           MOV  CR0,EAX

001D EB 01 90                           JMP  START1           ;cambio

0020                            START1:
0020 B8 0010                            MOV  AX,10H           ;selector 2
0023 8E D8                              MOV  DS,AX
0025 8E C0                              MOV  ES,AX
0027 8E D0                              MOV  SS,AX
0029 8E E8                              MOV  GS,AX
002B 8E E0                              MOV  FS,AX
002D 66| BC FFFFFFFF                    MOV  ESP,0FFFFFFFFH

                                ;en este momento nos encontramos en el modo protegido

0033                            CODE    ENDS
                                        END  START
```

Este es el sistema de modo protegido más sencillo posible cargando todos los registros del segmento, excepto código, con el mismo descriptor del segmento de datos de la GDT. El nivel de privilegio se inicializa a 00, el nivel más alto. Este sistema se utiliza con más frecuencia donde un usuario tiene acceso al microprocesador y requiere del espacio completo de la memoria.

En sistemas más complejos, los pasos que se requieren para inicializar el sistema en el modo protegido son más complicados. Para sistemas complejos que frecuentemente son multiusuario, los registros se cargan utilizando el segmento para el estado de tareas (TSS). A continuación siguen los pasos necesarios para colocar el 80386 en la operación del modo protegido para un sistema más complejo utilizando un cambio de tareas:

1. Inicialice la tabla del descriptor de interrupciones para que se refiera a interrupciones válidos con los 32 descriptores, por lo menos, en la IDT.
2. Inicialice la tabla de descriptores globales para que contenga por lo menos dos descriptores de segmento para el estado de tareas (TSS) y los segmentos iniciales de código y datos necesarios para la tarea inicial.
3. Inicialice el registro de tareas (TR) para que apunte a un TSS válido porque cuando ocurre la conmutación inicial de tareas y se accesa el nuevo TSS, los registros actuales se almacenan en el TSS inicial.
4. Cambie al modo protegido utilizando un brinco intrasegmento (cercano) para borrar la cola interna de instrucciones. Cargue el TR con el selector actual de TSS.
5. Cargue el TR con una instrucción de brinco lejano para direccionar el nuevo TSS y almacenar el estado actual.
6. El 80386 se encuentra operando ahora en el modo protegido bajo el control de la primera tarea.

El ejemplo 14-4 muestra el programa requerido para inicializar el sistema y cambiar al modo protegido utilizando una conmutación de tareas. La tarea inicial del sistema opera al nivel más alto de protección (00) y controla todo el ambiente operativo del 80386. En muchos casos, se utiliza para reinicializar (cargar) la programación que permita a muchos usuarios direccionar el sistema en un ambiente multiusuario.

EJEMPLO 14–4 (página 1 de 5)

```
0008                DESC     STRUC              ;define la estructura del descriptor

0000 0000           LIMIT_L  DW      0
0002 0000           BASE_L   DW      0
0004 00             BASE_M   DB      0
0005 00             ACCESS   DB      0
0006 00             LIMIT_H  DB      0
0007 00             BASE_H   DB      0

                    DESC     ENDS

0068                TSS      STRUC              ;define la estructura del TSS

0000 0000           BACK_L   DW      0
0002 0000                    DW      0
0004 00000000       ESP0     DD      0
```

EJEMPLO 14–4 (página 2 de 5)

```
0008 0000               SS0      DW      0
000A 0000                        DW      0
000C 00000000           ESP1     DD      0
0010 0000               SS1      DW      0
0012 0000                        DW      0
0014 00000000           ESP2     DD      0
0018 0000               SS2      DW      0
001A 0000                        DW      0
001C 00000000           CCR3     DD      0
0020 00000000           EIP      DD      0
0024 00000000           TFLAGS   DD      0
0028 00000000           EEAX     DD      0
002C 00000000           EECX     DD      0
0030 00000000           EEDX     DD      0
0034 00000000           EEBX     DD      0
0038 00000000           EESP     DD      0
003C 00000000           EEBP     DD      0
0040 00000000           EESI     DD      0
0044 00000000           EEDI     DD      0
0048 0020               EES      DW      20H
004A 0000                        DW      0
004C 0018               ECS      DW      18H
004E 0000                        DW      0
0050 0020               ESS      DW      20H
0052 0000                        DW      0
0054 0020               EDS      DW      20H
0056 0000                        DW      0
0058 0020               EFS      DW      20H
005A 0000                        DW      0
005C 0020               EGS      DW      20H
005E 0000                        DW      0
0060 0000               ELDT     DW      0
0062 0000                        DW      0
0064 0000                        DW      0
0066 0000               BIT_MAP  DW      0

                        TSS      ENDS

0000                    STACK    SEGMENT STACK

0000 0400 [                      DW      400H DUP (?)
           0000
               ]

0800                    STACK    ENDS

                        .386P
0000                    CODE     SEGMENT USE16
                                 ASSUME  CS:CODE,SS:STACK

0000 0000   0000 00000000        TSS1    TSS  <>          ;segmento 1 para el estado de tareas
     0000 0000 00000000
     0000 0000 00000000
     0000 0000 00000000
     00000000 00000000
     00000000 00000000
     00000000 00000000
     00000000 00000000
```

EJEMPLO 14–4 (página 3 de 5)

```
          00000000 00000000
          0020 0000 0018
          0000 0020 0000
          0020 0000 0020
          0000 0020 0000
          0000 0000 0000
          0000

0068 0000 0000 00000000         TSS2      TSS  <>              ;segmento 2 para el estado de tareas
          0000 0000 00000000
          0000 0000 00000000
          0000 0000 00000000
          00000000 00000000
          00000000 00000000
          00000000 00000000
          00000000 00000000
          00000000 00000000
          0020 0000 0018
          0000 0020 0000
          0020 0000 0020
          0000 0020 0000
          0000 0000 0000
          0000

00D0 0800 [              IDT      DB        8*256 DUP (?)      ;tabla para el descriptor de interrupciones
            00
             ]

08D0 0000 0000 00 00     GDT      DESC      <>                 ;descriptor nulo
     00 00
08D8 0000 0028 00 85     TGI      DESC      <0,28H,0,85H,0,0>  ;compuerta No. 1 de tareas
     00 00
08E0 0000 0030 00 85     TG2      DESC      <0,30H,0,85H,0,0>  ;compuerta No. 2 de tareas
     00 00
08E8 FFFF 0000 00 9A     CS       DESC      <-1,0,0,9AH,0CFH,0> ;segmento de código (4G)
     CF 00
08F0 FFFF 0000 00 92     DS1      DESC      <-1,0,0,92H,0CFH,0> ;segmento de datos (4G)
     CF 00
08F8 FFFF 0000 00 89     TSS1     DESC      <-1,0,0,89H,0CFH,0> ;TSS1 disponible
     CF 00
0900 FFFF 0000 00 89     TSS2     DESC      <-1,0,0,89H,0CFH,0> ;TSS2 disponible
     CF 00
0908 2000 [              IOBP     DB        2000H DUP (0)      ;habilitar todas las entradas/salidas
           00
             ]
2908 FF                           DB        0FFH               ;fin del mapa de bits de entrada/salida

0006                     GDT_A    STRUC

 0000 0000               A        DW        0
 0002 0000               B        DW        0
 0004 0000               CC       DW        0

                         GDT_A    ENDS

2909 0000 0000 0000      GDT_A    GDT_A     <>                 ;dirección de GDT

290F                     MAIN     PROC      FAR
```

EJEMPLO 14–4 (página 4 de 5)

```
290F 8C C8                  MOV    AX,CS
2911 8E D8                  MOV    DS,AX
2913 E8 0098                CALL   DO_IDT               ;inicializar IDT
2916 B8 0908 R              MOV    AX,OFFSET IOBP       ;inicializar mapas de bits de entrada/salida
2919 2E: A3 0066 R          MOV    TSS1.BIT_MAP,AX
291D 2E: A3 00CE R          MOV    TSS2.BIT_MAP,AX

2921 66| 33 C0              XOR    EAX,EAX              ;obtener dirección lineal de arranque
2924 8C C8                  MOV    AX,CS
2926 66| C1 E0 04           SHL    EAX,4
292A 66| 33 DB              XOR    EBX,EBX
292D BB 29AE R              MOV    BX,OFFSET TASK1
2930 66| 50                 PUSH   EAX
2932 66| 03 C3              ADD    EAX,EBX
2935 66| 2E: A3 0088 R      MOV    TSS2.EIP,EAX         ;inicializar TASK1 como dirección de arranque
293A 66| 58                 POP    EAX
293C 66| 50                 PUSH   EAX
293E BB 0000 R              MOV    BX,OFFSET TSS1       ;cargar la dirección TSS1
2941 66| 03 C3              ADD    EAX,EBX
2944 2E: A3 0002 R          MOV    TSS1.BASE_L,AX
2948 66| C1 E8 10           SHR    EAX,16
294C 2E: A2 0004 R          MOV    TSS1.BASE_M,AL
2950 2E: 88 26 0007 R       MOV    TSS1.BASE_H,AH
2955 66| 58                 POP    EAX
2957 66| 50                 PUSH   EAX
2959 BB 0068 R              MOV    BX,OFFSET TSS2       ;cargar la dirección TSS2
295C 66| 03 C3              ADD    EAX,EBX
295F 2E: A3 006A R          MOV    TSS2.BASE_L,AX
2963 66| C1 E8 10           SHR    EAX,16
2967 2E: A2 006C R          MOV    TSS2.BASE_M,AL
296B 2E: 88 26 006F R       MOV    TSS2.BASE_H,AH

2970 66| 58                 POP    EAX
2972 B8 FFFF                MOV    AX,-1
2975 2E: A3 290B R          MOV    GDT_AD.B,AX
2979 66| 58                 POP    EAX
297B 66| C1 E0 04           SHL    EAX,4
297F BB 08D0 R              MOV    BX,OFFSET GDT
2982 66| 03 C3              ADD    EAX,EBX
2985 2E: A3 2909 R          MOV    GDT_AD.A,AX
2989 66| C1 E8 10           SHR    EAX,16
298D 2E: A3 290D R          MOV    GDT_AD.CC,AX         ;inicializar la dirección GDT
2991 2E: 0F 01 16 2909 R    LGDT   GDT_AD               ;cargar el registro de GDT
2997 0F 20 C0               MOV    EAX,CR0              ;instalar PM
299A 66| 83 C8 01           OR     EAX,1
299E 0F 22 C0               MOV    CR0,EAX              ;modo protegido
29A1 EB 00                  JMP    NEXT

29A3               NEXT:

29A3 B8 0008                MOV    AX,8
29A6 0F 00 D8               LTR    AX                   ;dirección TSS1
29A9 B8 0010                MOV    AX,10H
29AC FF E0                  JMP    AX                   ;brincar a TSS2

29AE               MAIN     ENDP

29AE               DO_IDT   PROC   NEAR
```

EJEMPLO 14–4 (página 5 de 5)

```
                        ;set up interrupts (now shown here)

29AE                    DO_IDT  ENDP

29AE                    TASK1:                                  ;el programa principal empieza aqui

29AE                    CODE    ENDS
                        END     MAIN
```

14–6 MODO VIRTUAL 8086

Un modo especial de operación que hasta ahora no se ha discutido es el modo virtual 8086. Este modo especial está diseñado para que múltiples programas de aplicación para el modo real 8086 se puedan ejecutar simultáneamente. La figura 14-28 muestra dos aplicaciones del 8086 mapeadas en el 80386 utilizando el modo virtual. Cuando el sistema operativo permite que se ejecuten aplicaciones múltiples, generalmente se realizan a través de la técnica llamada *división del tiempo*. El sistema operativo designa una cantidad determinada de tiempo para cada tarea. Por ejemplo, si se están ejecutando tres tareas, el sistema operativo puede designar 1 ms a cada tarea. Esto significa que después de cada milésima de segundo, ocurre un cambio de tarea a la siguiente tarea. De esta manera todas las tareas reciben una parte del tiempo de ejecución del 80386, resultando en un sistema que parece ejecutar más de una tarea a la vez. Los tiempos de las tareas se pueden ajustar para darle a cualquier tarea cualquier porcentaje del tiempo de ejecución del microprocesador.

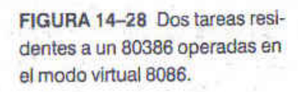
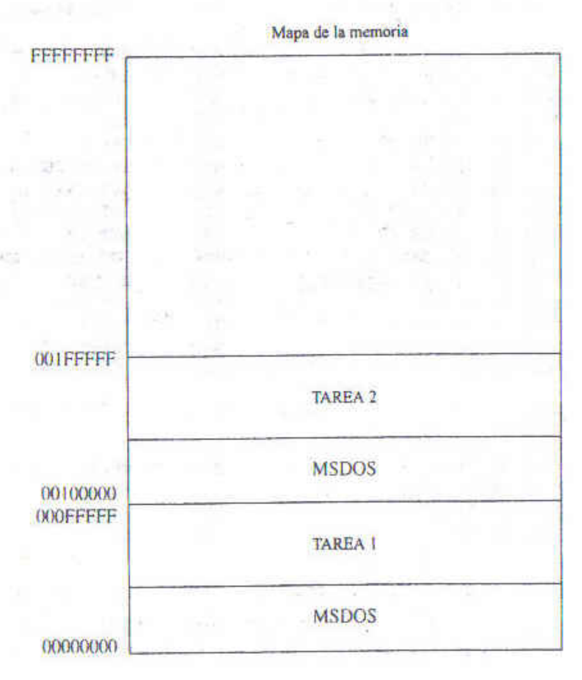


FIGURA 14–28 Dos tareas residentes a un 80386 operadas en el modo virtual 8086.



Un sistema que puede utilizar esta técnica es un área de impresión (spooler). Este puede funcionar en una fracción del DOS y se puede accesar en el 10 por ciento del tiempo. Esto permite que el sistema imprima utilizando el área de impresión, sin afectar el funcionamiento del sistema porque sólo se utiliza el 10 por ciento del tiempo del sistema.

La diferencia principal entre la operación del modo protegido 80386 y el modo virtual 8086 es la forma en que se interpretan los registros de segmentos del microprocesador. En el modo virtual 8086, los registros de segmentos se utilizan como en el modo real. O sea como una dirección de segmento y una dirección de desplazamiento capaz de accesar un espacio de memoria de 1M bytes desde la localidad 00000H-FFFFFH. El acceso a muchos sistemas en modo virtual 8086 se hace posible por la unidad de paginación que se explica en la siguiente sección. A través de la paginación, el programa aún accesa la memoria abajo del límite de 1M byte, sin embargo el microprocesador puede direccionar un espacio en la memoria física en cualquier localidad en el campo de 4G bytes del sistema de la memoria.

Se entra al modo virtual 8086 activando el bit VM en el registro de EFLAG con un 1 lógico. Se entra a este modo por medio de una instrucción de IRET si el nivel de privilegio es de 00. Esta terminal no se puede activar de ninguna otra manera. Intentar direccionar una dirección de la memoria arriba del límite de 1M bytes puede hacer que ocurra un interruptor de tipo 13.

El modo virtual 8086 se puede utilizar para compartir un microprocesador con muchos usuarios particionando la memoria, para que cada usuario tenga su propia partición del DOS. Al usuario 1 se le pueden asignar las localidades 00100000H-001FFFFFH de la memoria, al usuario 2 las localidades 00200000H-002FFFFFH, etcétera. El programa del sistema situado en las localidades 00000000H-000FFFFFH podría entonces compartir el microprocesador entre los usuarios cambiándose de uno a otro para ejecutar sus programas. De esta manera, un microprocesador se comparte por muchos usuarios.

14-7 EL MECANISMO DE PAGINACION DE LA MEMORIA

El mecanismo de paginación permite que cualquier dirección lineal (lógica), como la genere un programa, se coloque en cualquier página de la memoria física, según se genere en el mecanismo de paginación. Una **página de la memoria lineal** es una página que se direcciona con un selector y con un desplazamiento, ya sea en el modo real o protegido. Una **página de la memoria física** es una página que existe en alguna localidad de la memoria física real. Por ejemplo, la localidad 20000H de la memoria lineal se puede mapear en la localidad 30000H de la memoria física, o en cualquier otra localidad, con la unidad de paginación. Esto significa que una instrucción que accesa la localidad 20000H en realidad accesa la localidad 30000H.

Cada página de la memoria 80386 es de 4K bytes de longitud. La paginación permite que el programa del sistema se coloque en cualquier dirección física con el mecanismo de paginación. Se utilizan tres componentes en la transformación de la dirección de la página: el directorio de páginas, la tabla de páginas, y la verdadera página de la memoria física.

El directorio de páginas

El directorio de páginas contiene las localidades de hasta 1,024 tablas para la traducción de páginas. Cada tabla para la traducción de páginas transforma una dirección lógica a una dirección

física. El directorio de páginas se almacena en la memoria y se accesa por medio del registro de dirección de descriptores de página (CR3). El registro de control CR3 tiene la dirección base del directorio de páginas, que empieza en cualquier límite de 4K del sistema de la memoria. La instrucción MOV CR3,reg se utiliza para inicializar el registro CR3 para la paginación. En un sistema de modo virtual 8086, cada partición del DOS en el 8086 tendría su propio directorio de páginas.

El directorio de páginas contiene hasta 1,024 entradas y cada una tiene 4 bytes. El directorio de páginas en sí ocupa una página de memoria de 4K bytes. Cada entrada del directorio de páginas (véase la figura 14-29) transforma los 10 bits más a la izquierda de la dirección de la memoria. Esta porción de 10 bits de la dirección lineal se utiliza para localizar diferentes tablas de páginas para diferentes entradas a tablas de página. La dirección para la tabla de páginas (A32-A12), almacenada en la entrada del directorio de páginas, accesa una tabla para la transformación de páginas de 4K bytes de largo. Para transformar por completo cualquier dirección lineal a cualquier dirección física, se necesitan 1,024 tablas de páginas, cada una tiene una longitud de 4K bytes más el directorio de la tabla de páginas, que es de 4K bytes de longitud. Este esquema de traducción requiere de hasta 4M más 4K bytes de memoria para una transformación completa de las direcciones. Solamente los sistemas operativos más grandes soportan este tamaño de transformación de direcciones. Muchos sistemas operativos encontrados comúnmente transforman solamente los primeros 16M o 32M bytes del sistema de la memoria si está habilitada la paginación. Esto incluye programas como Windows 3.1.[3]

Los bits de control de la entrada del directorio de la tabla de páginas, como se muestra en la figura 14-29, realizan las siguientes funciones:

1. D — Sin Definir (dirty): no están definidas las entradas del directorio de la tabla de páginas para el microprocesador 80386 y se proporciona para uso del sistema operativo.
2. A — Accesado: se activa con un 1 lógico cada vez que el microprocesador accesa la entrada del directorio de páginas.
3. R/W y U/S — Lectura/Escritura y Usuario/Supervisor: ambos se utilizan en el esquema de protección indicado en la tabla 14-5. Ambos bits se combinan para desarrollar la protección del nivel de prioridad de la paginación para el nivel 3, el nivel más bajo del usuario.
4. P — Presente: si es un 1 lógico, indica que la entrada se puede utilizar en la transformación de direcciones. Si P = 0, la entrada no se puede utilizar para la transformación. Una entrada no presente se puede utilizar para otros propósitos, como indicar que la página está almacenada actualmente en el disco. Si P = 0, los bits restantes de la entrada se pueden utilizar para indicar la localización de la página en el sistema de memoria de disco.

Tabla de páginas

La tabla de páginas contiene 1,024 direcciones de páginas físicas para transformar una dirección lineal a una dirección física. El formato para la entrada de la tabla de páginas es exactamente igual que la entrada para el directorio de páginas (véase la figura 14-29). La diferencia principal es que la entrada para el directorio de páginas contiene la dirección física de una tabla de páginas, mientras que la entrada para la tabla de páginas contiene la dirección física de una página de 4K

[3]Windows es un programa fabricado por Microsoft Corporation.

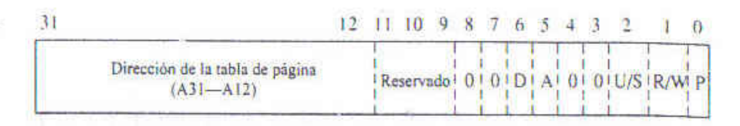


FIGURA 14-29 Entrada al directorio de la tabla de páginas.

TABLA 14-5 Protección para el nivel 3 utilizando U/S y R/W

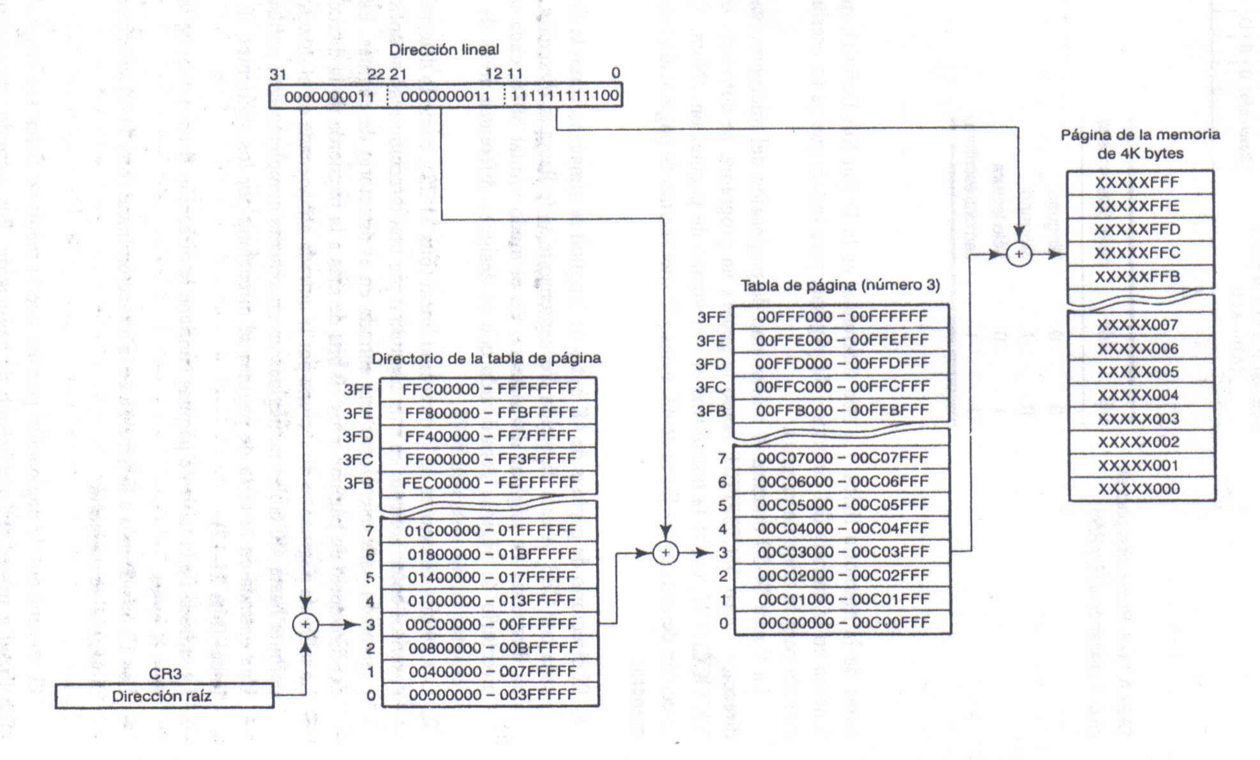
U/S	*R/W*	*Nivel 3 de Acceso*
0	0	Ninguno
0	1	Ninguno
1	0	Sólo lectura
1	1	Lectura/escritura

bytes de la memoria física. La otra diferencia es la D (bit Sin Definir), que no tiene ninguna función en la entrada del directorio de páginas, pero indica que se ha escrito una página de una entrada para la tabla de páginas.

La figura 14-30 muestra el mecanismo de paginación del microprocesador 80386. Aquí, la dirección lineal 00C03FFCH, como la genera un programa, se convierte en la dirección física XXXXX3FCH, según la transforma el mecanismo de paginación. (Nota: XXXXX es cualquier dirección de una página física de 4K bytes.) El mecanismo de paginación funciona de la siguiente manera:

1. El directorio de páginas de 4K bytes de longitud se almacena como la dirección física localizada por CR3. A esta dirección frecuentemente se le llama la *dirección raíz*. Existe a la vez un directorio de páginas del sistema. En el modo virtual 8086, cada tarea tiene su propio directorio de páginas permitiendo que se designen diferentes áreas de la memoria física a distintas tareas virtuales 8086.
2. Los 10 bits superiores de la dirección lineal (bits 31-22), como lo determinan los descriptores descritos anteriormente en este capítulo o por una dirección real, se aplican al mecanismo de paginación para seleccionar una entrada en el directorio de páginas. Esto mapea la entrada del directorio de páginas a los 10 bits de más a la izquierda de la dirección lineal.
3. La tabla de páginas se direcciona por la entrada almacenada en el directorio de páginas. Esto permite hasta 4K tablas de páginas en un sistema completamente poblado y transformado.
4. Una entrada en la tabla de páginas se direcciona por los siguientes 10 bits de la dirección lineal (bits 21-12).
5. La entrada de la tabla de páginas contiene la dirección física actual de la página de memoria de 4 K bytes.
6. Los 12 bits de más a la derecha de la dirección lineal (bits 11-0) seleccionan una localidad de la página de memoria.

El mecanismo de paginación permite que la memoria física sea asignada a cualquier dirección lineal a través del mecanismo de paginación. Por ejemplo, supongamos que la dirección lineal de 20000000H se selecciona en un programa, pero esta localidad de la memoria no existe en el sistema de la memoria física. La página lineal de 4K bytes será referenciada como las



Nota: 1. El intervalo de las direcciones ilustrado en el directorio de páginas y en la tabla de páginas representan el intervalo seleccionado de la dirección lineal y no del contenido de estas tablas.

2. Las direcciones (XXXXX) indicadas en la página de la memoria son seleccionadas por la entrada de la tabla de páginas.

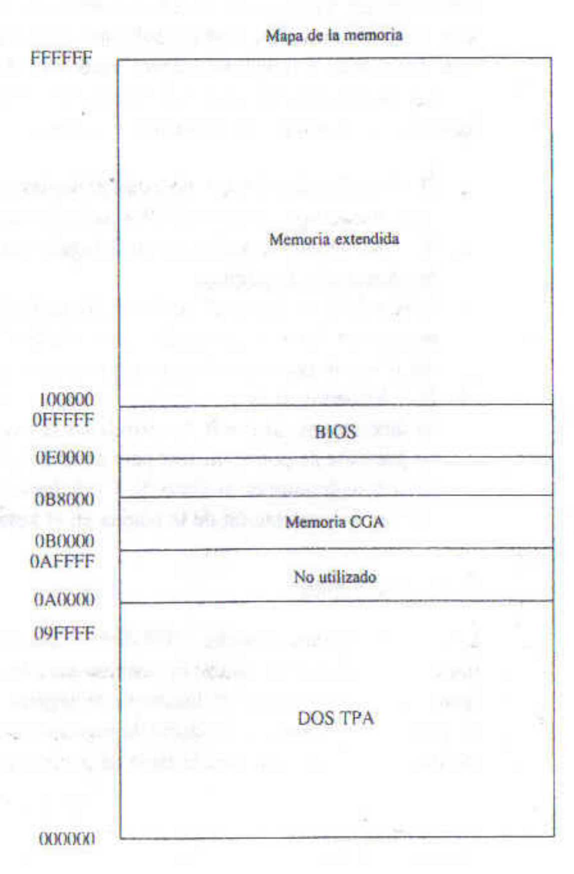
FIGURA 14–30 Transformación de la dirección lineal 00C03FFC a la dirección XXXXXFFC de la memoria física. El valor de XXXXX se determina por la entrada de la tabla de páginas (no se muestra aquí).

localidades 20000000H-20000FFFH por el programa. Debido a que esta sección de la memoria física no existe, el sistema operativo puede asignar una página existente de la memoria física como 12000000H-12000FFFH a este intervalo de la dirección lineal.

En el proceso para la transformación de direcciones, los 10 bits más a la izquierda de la dirección lineal seleccionan la entrada 200H del directorio de páginas localizada en la dirección de desplazamiento 800H del directorio de páginas. Esta entrada del directorio de página contiene la dirección de la tabla de páginas para las direcciones lineales 20000000H-203FFFFFH. Los bits (21-12) de la dirección lineal seleccionan una entrada de esta tabla de páginas que corresponde a una página de la memoria de 4K bytes. Para las direcciones lineales 2000000H-20000FFFH, se selecciona la primera entrada (la No. 0) de la tabla de páginas. Esta primera entrada contiene la dirección física de la página real de la memoria o 12000000H-12000FFFH en este ejemplo.

Considere, por ejemplo, un sistema de computación típico utilizando el DOS. El mapa de la memoria para el sistema aparece en la figura 14-31. Observe que en el mapa hay áreas de la memoria sin utilizar que se podrían paginar a una localidad diferente, dándole a un programa de aplicación en modo real bajo DOS más memoria. El sistema normal de la memoria del DOS empieza en la localidad 00000H y se extiende a la localidad 9FFFFH, que es de 640K bytes de

FIGURA 14-31 Mapa de la memoria para una máquina de estilo AT compatible.



memoria. Arriba de la localidad 9FFFFH se encuentran secciones dedicadas a las tarjetas de video, tarjetas de disco, y al BIOS ROM del sistema. En este ejemplo una área de la memoria justo arriba de 9FFFFH está sin utilizar (A0000-AFFFFH). Esta sección de la memoria se podría utilizar por el DOS para que el área total de la memoria para aplicaciones sea de 704K en lugar de 640K.

Esta sección de la memoria se puede utilizar mapeándola en la memoria extendida en las localidades 102000H-11FFFFH. El programa para realizar esta transformación e inicializar el directorio para la tabla de páginas y las tablas de páginas requeridas para instalar la memoria se ilustran en el ejemplo 14-5. Observe que este procedimiento inicializa el directorio para la tabla de páginas, una tabla de páginas, y carga el CR3. No se cambia al modo protegido y no activa la paginación. Observe que la paginación funciona en la operación de la memoria en modo real.

EJEMPLO 14–5

```
                          .386P
                          ;Programa que inicializa un sistema de memoria paginada
                          ;que accese la memoria que está arriba de la localidad FFFFFH.
                          ;
                          ;Directorio de página

0000                      DATA    SEGMENT

0000 = 0000               PG_DIR  EQU     $

0000 00001000 R           TAB0PT  DD      TAB0
0004 03FF [                       DD      1023 DUP (?)
               00000000
                        ]

                          ;Page Table

1000 0400 [               TAB0    DD      1024 DUP (?)
               00000000
                        ]

2000                      DATA    ENDS

0000                      CODE    SEGMENT USE16

                                  ASSUME  CS:CODE,DS:DATA

0000                      PAGES   PROC    FAR

0000 1E                           PUSH    DS
0001 06                           PUSH    ES
0002 FC                           CLD
0003 B8 ---- R                    MOV     AX,DATA           ;cargar los registros de segmento
0006 8E D8                        MOV     DS,AX
0008 8E C0                        MOV     ES,AX
000A BF 0000 R                    MOV     DI,OFFSET PG_DIR  ;direccionar tabla de página
000D 66| 83 05 07                 ADD     DWORD PTR [DI] ,7
0011 B9 0100                      MOV     CX,256
0014 BF 1000 R                    MOV     DI,OFFSET TAB0
0017 66| B8 00000007              MOV     EAX,7
```

```
001D                PAGES1:

001D 66| AB                  STOSD                          ;llenar la tabla de página
001F 66| 05 00001000         ADD     EAX,4096
0025 E2 F6                   LOOP    PAGES1
0027 BF 1280 R               MOV     DI,OFFSET TAB0+4*0A0H
002A 66| B8 00102007         MOV     EAX,00102007H
0030 B9 0010                 MOV     CX,16

0033                PAGES2:

0033 66| AB                  STOSD                          ;remapear A0000H-AFFFFH
0035 66| 05 00001000         ADD     EAX,4096               ;a 102000H-11FFFFH
003B E2 F6                   LOOP    PAGES2

003D 66| 33 C0               XOR     EAX,EAX
0040 8C D8                   MOV     AX,DS
0042 66| C1 E0 04            SHL     EAX,4
0046 0F 22 D8                MOV     C3,EAX;address page directory

0049 07                      POP     ES
004A 1F                      POP     DS
004B CB                      RET

004C                PAGES    ENDP

004C                CODE     ENDS

                             END
```

14–8 INTRODUCCION AL MICROPROCESADOR 80486

El microprocesador 80486 es un dispositivo altamente integrado que contiene más de 1,200,000 transistores. Localizados dentro de este poderoso circuito integrado hay una unidad para la administración de la memoria (MMU); un coprocesador numérico completo que es compatible con el 80387; una memoria caché de alta velocidad que contiene 8K bytes de espacio; y un microprocesador completo de 32 bits que es compatible hacia arriba con el microprocesador 80386. El 80486 está disponible actualmente en versiones de 25 MHz, 33 MHz y 50 MHz. Intel ha mostrado una versión del 80486 de 100 MHz, pero aún no sale al mercado. El 80486 viene como un 80486DX o como un 80486SX. La única diferencia entre estos dispositivos es que el 80486SX no contiene un coprocesador numérico, lo cual reduce el precio. El coprocesador numérico 80487SX está disponible como un componente separado para el microprocesador 80486SX. También están disponibles las versiones de doble reloj como el 80486DX2 (versiones de 50 MHz y 66 MHz). Las versiones de doble reloj operan internamente a 50 MHz o a 66 MHz, sin embargo utilizan una velocidad de canal de 25 MHz o 33 MHz para facilitar los requerimientos al sistema de la memoria. La versión de doble reloj de 50 MHz ejecuta los programas a una velocidad promedio entre las versiones de 33 MHz y 50 MHz. La versión de doble reloj de 66 MHz opera a una velocidad ligeramente mejor que la versión de 50 MHz. Observe que usualmente no se requiere de ningún cambio en el sistema para escalar a una versión de doble reloj en la mayoría de las tarjetas madre.

También están disponibles las versiones extendidas (Overdrive[4]) que son circuitos extras que se enchufan en una base junto al microprocesador, para incrementar el funcionamiento a casi lo mismo que la versión de doble reloj. El procesador Overdrive es una manera eficiente para escalar un microprocesador 80486SX si lo soporta la tarjeta madre.

Esta sección detalla las diferencias entre los microprocesadores 80486 y 80386. Estas diferencias son pocas; las diferencias más notables se aplican al sistema de memoria caché y al generador de paridad.

Diagrama de base de los microprocesadores 80486DX y 80486SX

La figura 14-32 presenta el diagrama de base del microprocesador 80486DX, para un encapsulado PGA de 168 terminales. El 80486SX, también está encapsulado en un PGA de 168 terminales, no se muestra porque existen pocas diferencias. Observe que la terminal B15 es NMI en el 80486DX y la A15 es NMI en el 80486SX. Las únicas otras diferencias son que la terminal A15 es $\overline{\text{IGNNE}}$ en el 80486DX (no presente en el 80486SX), la C14 es $\overline{\text{FERR}}$ en el 80486DX y las terminales B15 y C14 en el 80486SX no están conectadas.

Cuando se conecta el microprocesador 80486, todas las terminales de VCC y VSS deben estar conectados a la fuente de alimentación para un funcionamiento correcto. La fuente de voltaje debe ser capaz de suministrar 5.0 V ± 10%, con un consumo de hasta 1.2 A de corriente para la versión de 33 MHz. El consumo promedio de corriente es de 650 mA para la versión de 33 MHz. Un cero lógico de salida permite hasta 4.0 mA de corriente y un uno lógico de salida hasta 1.0 mA. Si se requieren corrientes más grandes, como a menudo lo son, entonces el 80486 debe ser acoplado. La figura 14-33 muestra un sistema 80486DX con acoplamiento. En el circuito mostrado, sólo las señales de dirección, datos y paridad están acopladas.

Definiciones de las terminales

1. A31-A2 (Salida de direcciones): proporcionan a la memoria y a E/S con la dirección durante el funcionamiento normal y durante la invalidación de una línea caché, A31-A4 son utilizados para manejar al microprocesador.
2. $\overline{\text{A20M}}$ (Máscara de dirección de sólo 20 bits): usado para ocasionar que el 80486 "enrolle" la dirección de la localidad 000FFFFFH a la 00000000H como lo hace el microprocesador 8086. Esto proporciona un sistema de memoria que funciona como lo hace el 1M byte de memoria en el microprocesador 8086. La mayoría de los sistemas no utilizan el enmascaramiento de direcciones porque el programa HIMEM.SYS no puede tener acceso a la memoria adicional ubicada en las direcciones 100000H-10FFEFH.
3. ADS (Habilitación de dirección de dato): se convierte en un cero lógico para indicar que el canal de direcciones contiene una dirección de memoria válida.
4. AHOLD (Entrada de solicitud del canal de direcciones): causa que el microprocesador coloque sus conexiones del canal de direcciones en su estado de alta impedancia, con el resto de los canales quedando activos. Comúnmente utilizado por otro amo de canal para ganar el acceso para un ciclo de invalidación del caché.

[4]Overdrive es una marca registrada de Intel Corporation.

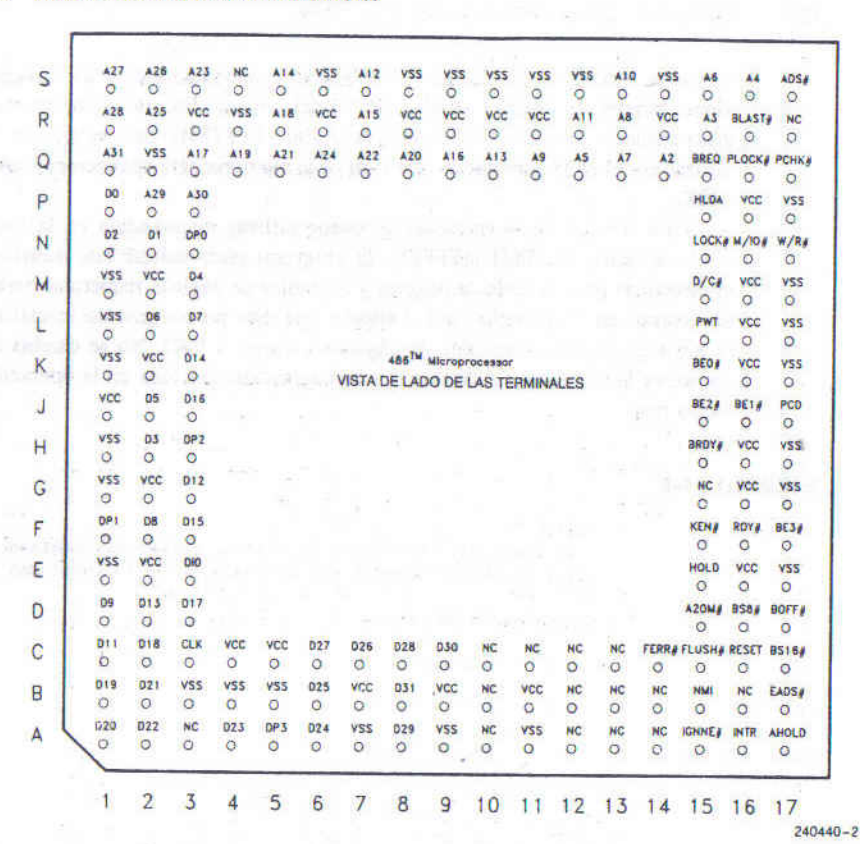


FIGURA 14–32 Diagrama de base del 80486. (Cortesía de Intel Corporation.)

5. $\overline{BE3}$ –$\overline{BE1}$ (Habilitación de salidas byte): selecciona un banco del sistema de la memoria cuando la información es transferida entre el microprocesador, y su espacio de memoria y de E/S. La señal $\overline{BE3}$ habilita a D31-D24, $\overline{BE2}$ habilita a D23-D16, $\overline{BE1}$ habilita a D15-D8 y $\overline{BE0}$ habilita a D7-D0.
6. $\overline{BLAST}$ (Ultima salida en ráfaga): muestra que el ciclo ráfaga del canal se completa en la próxima habilitación de la señal $\overline{BRDY}$.
7. $\overline{BOFF}$ (Entrada de deshabilitación de canales): ocasiona que el microprocesador coloque sus canales en el estado alta impedancia durante el próximo ciclo del reloj. El microprocesador permanece en el estado de cesión del canal hasta que la terminal $\overline{BOFF}$ es un nivel 1 lógico.
8. $\overline{BRDY}$ (Entrada de ráfaga lista): utilizado para señalarle al microprocesador que se completó un ciclo ráfaga.
9. BREQ (Salida de solicitud de canal): indica que el 80486 ha generado una solicitud interna del canal.
10. $\overline{BS8}$ (Canal de entrada de información de tamaño 8): causa que el 80486 se autoestructure con un canal de datos de 8-bits para tener acceso a memoria y componentes E/S de ancho byte.

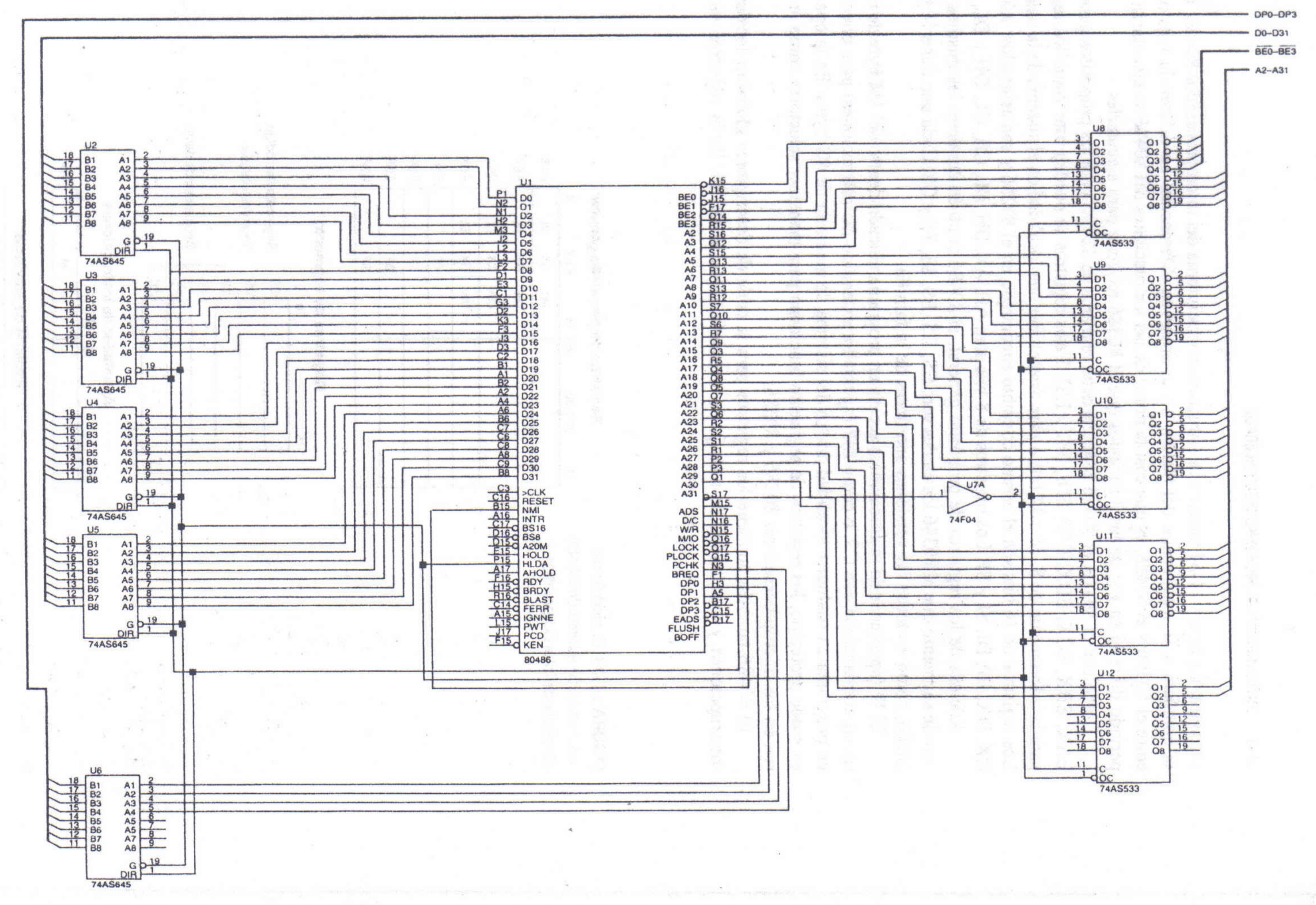


FIGURA 14–33 Un microprocesador 80486 que muestra los canales de direcciones, datos y paridad acoplados.

11. $\overline{\text{BS16}}$ (Canal de entrada de tamaño 16): causa que el 80486 se autoestructure con un canal de datos de 16 bits para tener acceso a memoria y componentes E/S de ancho de palabra.
12. CLK (Entrada de reloj): proporciona al 80486 la señal de temporización básica. La entrada de reloj es una entrada TTL que es de 25 MHz para operar el 80486 a 25 MHz.
13. D31 –D0 (Canal de datos): transfiere datos entre el microprocesador, la memoria y el sistema de E/S. Las conexiones del canal de datos D7–D0 también son utilizadas para aceptar el número de vector de interrupción durante un ciclo de reconocimiento de interrupción.
14. D/$\overline{\text{C}}$ (Datos/$\overline{\text{Control}}$): indica si la operación actual es una transferencia de datos o un ciclo de control. Consulte la tabla 14-6 para la función de D/$\overline{\text{C}}$,M/$\overline{\text{IO}}$ y W/$\overline{\text{R}}$.
15. DP3–DP0 (Paridad de datos de E/S de información): proporcionan paridad para una operación de escritura y verifican la paridad para una operación de lectura. Si un error de paridad es detectado durante una lectura, la salida $\overline{\text{PCHK}}$ se convierte en un 0 lógico para indicar un error de paridad. Si la paridad no es utilizada en un sistema, estas líneas deben ser llevadas al nivel alto de +5.0 V.
16. $\overline{\text{EADS}}$ (Entrada de habilitación estroboscópica de dirección externa): utilizado con AHOLD para señalar que una dirección externa se utiliza para realizar un ciclo de invalidación caché.
17. $\overline{\text{FERR}}$ (Salida de error de punto flotante): indica que el coprocesador de punto flotante ha detectado una condición de error. Utilizado para mantener compatibilidad con el DOS.
18. $\overline{\text{FLUSH}}$ (Entrada para eliminar contenido del caché): fuerza al microprocesador a borrar el contenido de su caché interno de 8K bytes.
19. HLDA (Salida para reconocer la cesión de canal [Hold]): indica que la entrada HOLD está activa y que el microprocesador ha colocado sus canales en su estado de alta impedancia.
20. HOLD (Entrada de solicitud de canal): utilizado para solicitar una acción de DMA. Causa que los canales de direcciones, datos y control sean llevados a su estado de alta impedancia y además, que una vez cedidos HLDA se convierta en un 0 lógico.
21. $\overline{\text{IGNNE}}$ (Ignorar la entrada de error numérico): ocasiona que el microprocesador ignore los errores de punto flotante y continúe procesando datos. Esta señal no afecta el estado de la terminal $\overline{\text{FERR}}$.
22. INTR (Entrada de solicitud de interrupción): solicita una interrupción enmascarable como lo hacen todos los otros miembros de la familia.
23. KE$\overline{\text{N}}$ (Entrada de habilitación del caché): causa que el canal actual se almacene en el caché interno.
24. $\overline{\text{LOCK}}$ (Salida): se convierte en un 0 lógico para cualquier instrucción que tenga el prefijo LOCK.

TABLA 14–6 Identificación de ciclo de canales

M/$\overline{IO}$	*D/$\overline{C}$*	*W/$\overline{R}$*	*Tipo de ciclo de canal*
0	0	0	Reconocimiento de interrupción
0	0	1	Alto/especial
0	1	0	Leer E/S
0	1	1	Escribir E/S
1	0	0	Ciclo de código (fetch)
1	0	1	Reservado
1	1	0	Lectura de memoria
1	1	1	Escritura de memoria

25. M/$\overline{\text{IO}}$ (Memoria/$\overline{\text{IO}}$): define si el canal de datos contiene una dirección de memoria o un número de puerto de E/S. También se combina con la señal W/$\overline{\text{R}}$ para generar las señales para control de lectura y escritura a memoria y E/S.
26. NMI (Entrada de interrupción no enmascarable): ocasiona una interrupción de tipo 2.
27. PCD (Salida de deshabilitación de página de caché): refleja el estado del bit de atributo PCD en la entrada de la tabla de páginas o la entrada del directorio de páginas.
28. $\overline{\text{PCHK}}$ (Salida de verificación de paridad): indica que se detectó un error de paridad durante una operación de lectura en las terminales DP3–DP0.
29. $\overline{\text{PLOCK}}$ (Salida pseudobloqueada): indica que la operación actual requiere de más de un ciclo de canal para ejecutarse. Esta señal se convierte en un 0 lógico para las operaciones del coprocesador aritmético que accesa datos de la memoria de 64 u 80 bits.
30. PWT (Escritura de página a través de la salida): indica el estado del bit de atributo PWT en la entrada de la tabla de páginas o la entrada del directorio de páginas.
31. $\overline{\text{RDY}}$ (Entrada READY): indica que un ciclo no ráfaga del canal está completo. La señal $\overline{\text{RDY}}$ se debe regresar o el microprocesador coloca estados de espera en su temporización hasta que $\overline{\text{RDY}}$ se active.
32. RESET (Entrada de reinicialización): reinicializa al 80486 como lo hace en los otros miembros de la familia. La tabla 14-7 muestra el efecto de la entrada RESET en el microprocesador 80486.
33. W/$\overline{\text{R}}$ (Escribir/$\overline{\text{Leer}}$): señala que el ciclo del canal actual es para leer o escribir.

Arquitectura básica del 80486

La arquitectura del 80486DX es casi idéntica a la del 80386 más el coprocesador matemático 80387 y un caché interno de 8K bytes. El 80486SX es casi idéntico a un 80386 con un caché de

TABLA 14–7 El efecto de la señal de RESET

Registro	*Valor inicial con la auto-comprobación*	*Valor inicial sin la auto-comprobación*
EAX	00000000H	?
EDX	00000400H+ID[1]	00000400H+ID
EFLAGS	00000002H	00000002H
EIP	0000FFF0H	0000FFF0H
ES	0000H	0000H
CS	F000H	F000H
DS	0000H	0000H
SS	0000H	0000H
FS	0000H	0000H
GS	0000H	0000H
IDTR	Base = 0, límite 3FFH	Base = 0, límite 3FFH
CR0	60000010H	60000010H
DR7	00000000H	00000000H

[1]Numero de revisión del ID proporcionado por Intel para revisiones al microprocesador.

8K bytes. La figura 14-34 muestra la estructura básica interna del microprocesador 80486. Si esto se compara a la arquitectura del 80386, no se observan diferencias. La diferencia más notable entre el 80386 y el 80486, es que casi la mitad de las instrucciones del 80486 se ejecutarán en un periodo de reloj en vez de los 2 periodos que el 80386 requiere para ejecutarles.

Como con el 80386, el 80486 contiene ocho registros de 32 bits para los propósitos generales: EAX, EBX, ECX, EDX, EBP, EDI, ESI y ESP. Estos registros se pueden usar como los registros para la información de 8, 16 o 32 bits o para direccionar una localidad en el sistema de la memoria. Los registros de 16 bits son el mismo conjunto encontrado en el 80286 y son asignados: AX, BX, CX, DX, BP, DI, SI y SP. Los registros de 8 bits son: AH, AL, BH, BL, CH, CL, DH y DL.

Además de los registros de propósito general, el 80486 también contiene los mismos registros de segmento que el 80386 los cuales son: CS, DS, ES, SS, FS y GS. Cada uno tiene 16 bits de ancho, como en todas las versiones anteriores de la familia.

El IP (apuntador de instrucciones) accesa un programa ubicado dentro del 1M byte de memoria en combinación con CS, o como EIP (apuntador extendido de instrucciones) para direccionar un programa en cualquier localidad dentro del sistema de memoria de 4G bytes. En la operación en modo protegido, los registros de segmento funcionan para mantener selectores como lo hicieron en los microprocesadores 80286 y 80386.

El 80486 también contiene los registros para la tabla de descriptores globales, locales y de interrupciones y una unidad de manejo de memoria como el 80386. Estos registros no están

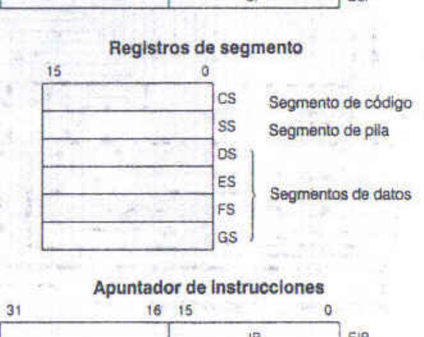


FIGURA 14-34 El modelo de programación interna del 80486. (Cortesía de Intel Corporation.)

mostrados en la figura 14-34, pero están presentes en el 80386. La función de la MMU y su unidad de paginación se describió con el 80386 anteriormente en este capítulo.

El registro extendido de banderas (EFLAGS) se muestra en la figura 14-35. Como en los otros miembros de la familia, los bits de la bandera más a la derecha realizan las mismas funciones para compatibilidad. A continuación se da una lista de cada bit de bandera con una descripción de su función.

1. AC (Verificación de alineación): nueva para el microprocesador 80486, utilizada para indicar que el microprocesador ha tenido acceso a una palabra en una dirección de paridad non o una doble palabra almacenada en un límite que no es de doble palabra. Para una ejecución más eficiente de los programas se requiere que la información se almacene en límites de palabra o doble palabra.
2. VM (Modo virtual): se activa este bit mientras que el 80486 se opere en el modo protegido. Este bit siempre será desactivado por una instrucción PUSHF, aun si el 80486 está trabajando en el modo virtual.
3. RF (Resume): utilizado en conjunto con los registros de depuración, como fue definido anteriormente en el capítulo para el microprocesador 80386.
4. NT (Tarea anidada): se activa para indicar que el 80486 está realizando una tarea que está anidada dentro de otra tarea.
5. IOPL (Nivel de privilegio de E/S): indica el nivel de privilegio máximo actual asignado al sistema de E/S.
6. OF (Sobreflujo): indica que el resultado de una operación aritmética con signo ha rebasado la capacidad del destino. También se utiliza con la instrucción de multiplicación.
7. DF (Dirección): selecciona una operación de autoincremento (DF = 0) o autodecremento (DF = 1) para las instrucciones de cadenas.
8. IF (Habilitación de interrupciones): habilita la terminal INTR si este bit está activo.

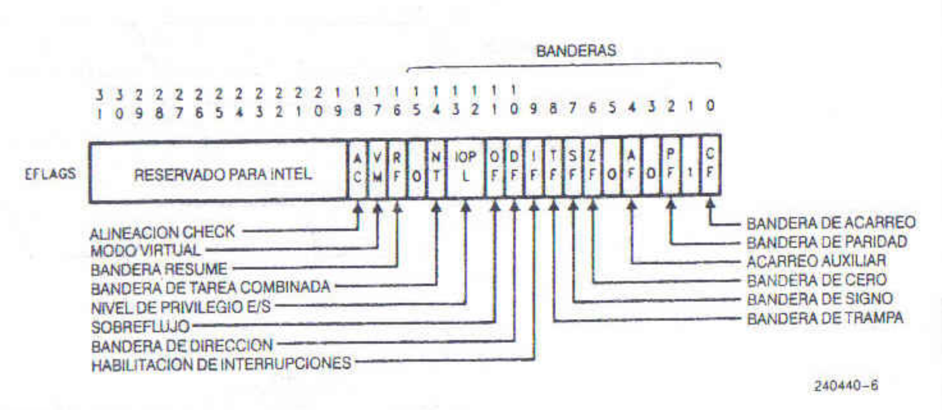


FIGURA 14-35 El registro EFLAG del 80486. (Cortesía de Intel Corporation.)

9. TF (Trampa): activado para habilitar depuración como fue descrito en los registros de depuración.
10. SF (Signo): indica que el signo del resultado está activo o desactivado.
11. ZF (Cero): indica que el resultado de una operación aritmética o lógica es cero (ZF = 1) o no es cero (ZF = 0).
12. AF (Auxiliar): utilizado con las instrucciones DAA y DAS para ajustar el resultado de una suma o resta BCD.
13. PF (Paridad): indica la paridad del resultado de una operación aritmética o lógica. Si la paridad es impar, PF = 0, y si la paridad es par, PF = 1.
14. CF (Acarreo): muestra si ocurrió un acarreo después de una suma o un préstamo después de una resta.

Sistema de memoria del 80486

El sistema de la memoria del 80486 es idéntico al del microprocesador 80386. El 80486 contiene 4G bytes de memoria comenzando en la localidad 00000000H y terminando en la localidad FFFFFFFFH. El cambio principal en el sistema de memoria es interno en el 80486, en la forma de una memoria para caché de 8K bytes la cual acelera la ejecución de instrucciones y la adquisición de información. Otra adición es el verificador/generador de paridad incluido en el microprocesador 80486.

Verificador/generador de paridad. La *paridad* es frecuentemente usada para determinar si la información se leyó correctamente de una localidad de la memoria. Para facilitar esto, Intel ha incorporado un *verificador/generador de paridad* interno. La paridad se genera en el 80486 durante cada ciclo de escritura. La paridad será generada como *paridad par* y un bit de paridad será proporcionado para cada byte de la memoria. Los bits para la verificación de la paridad aparecen en las terminales DP0–DP3, las cuales también son entradas así como salidas de paridad. Estos son normalmente archivados en la memoria durante cada ciclo de escritura y leídos de la memoria durante cada ciclo de lectura.

En una lectura, el microprocesador revisa la paridad y genera un error de revisión de paridad, *si esto ocurre,* en la terminal $\overline{\text{PCHK}}$. Un error de paridad no causa ningún cambio en el procesamiento a menos que el usuario aplique la señal $\overline{\text{PCHK}}$ a una entrada de interrupción. Las interrupciones son utilizadas frecuentemente para señalar un error de paridad en los sistemas de computadora que utilizan DOS. La figura 14-36 muestra la organización del sistema de la memoria del 80486, que incluye almacenamiento para la paridad. Observe que éste es el mismo del 80386, excepto por el almacenamiento del bit de paridad. Si la paridad no se utiliza, Intel recomienda que las terminales DP0–DP3 sean llevadas a +5.0 V.

Memoria caché. El sistema de la memoria caché almacena los datos utilizados por un programa y también las instrucciones. El *caché* está organizado como un conjunto asociativo de cuatro vías con cada localidad (línea) conteniendo 16 bytes o cuatro dobles palabras de información. El caché funciona en el modo de escritura a *través del caché.* Observe que el caché sólo cambia si ocurre una falla. Esto significa que los datos escritos a una localidad de la memoria no están capturados hasta que no se escriban al caché. En algunos casos, mucha de la porción activa de un programa se encuentra completamente dentro de la memoria caché. Esto ocasiona que la ejecución ocurra a una velocidad de 1 ciclo de reloj para muchas de las instrucciones que son utilizadas más comúnmente en un programa. Casi la única manera de que estas instrucciones eficientes sean

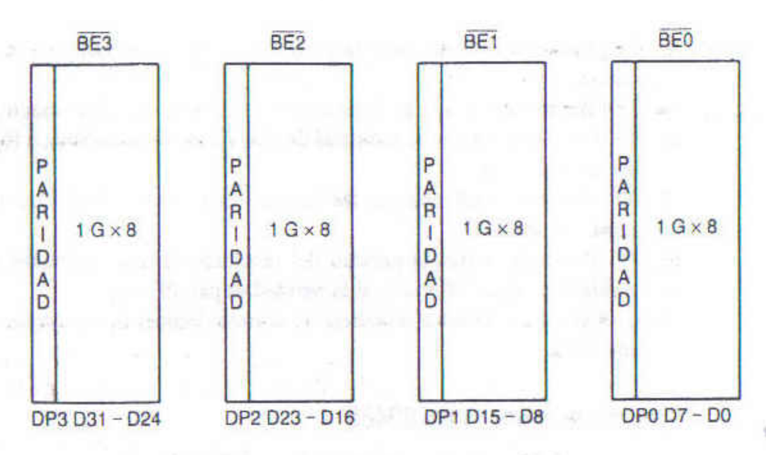


FIGURA 14-36 La organización de la memoria del 80486 mostrando paridad.

retrasadas es cuando el microprocesador debe llenar una línea en el caché. Los datos también se almacenan en el caché, pero tienen menos impacto sobre la velocidad de ejecución de un programa, porque la información no se referencia tan repetidamente como lo son muchas porciones de un programa.

El registro para control 0 (CR0) es utilizado para controlar al caché con dos bits nuevos de control no presentes en el microprocesador 80386. (Consulte la figura 14-37 para CR0 en el microprocesador 80486.) Los bits CD (*deshabilitar caché*) y NW (*escritura no a través del caché*) son nuevos para el 80486 y son utilizados para controlar el caché de 8K bytes. Si el bit CD es un 1 lógico, todas las operaciones del caché son inhibidas. Esta función sólo será usada para depuración de programas y normalmente permanece inactivo. El bit NW se usa para inhibir una operación de escritura directa al caché. Como con el CD, el caché a través de escritura es sólo inhibido para realizar pruebas. Para el funcionamiento normal de un programa, CD = 0 y NW = 0.

Debido a que el caché es nuevo en el microprocesador 80486 y el caché se llena usando ciclos de ráfaga no presentes en el 80386, se necesita de algunos detalles para entender los ciclos de llenado del canal. Cuando una línea del canal está llena, el 80486 deberá adquirir cuatro números de 32 bits del sistema de la memoria para llenar una línea en el caché. El llenado se logra con un *ciclo de ráfaga*. El ciclo de ráfaga es una memoria especial en donde cuatro números de 32 bits se recuperan del sistema de la memoria en 5 periodos del reloj. Esto asume que la velocidad de la memoria es suficiente y que no se requiere ningún estado de espera. Si la frecuencia del reloj del 80486 es de 33 MHz podemos llenar una línea caché en 167 ns, lo cual es muy eficiente considerando que una operación normal de lectura de la memoria normal, no ráfaga de 32 bits requiere de 2 periodos de reloj.

Temporización de la lectura de memoria. La figura 14-38 muestra la temporización de lectura del 80486 para una operación no ráfaga a la memoria. Observe que dos periodos de reloj son usados para transferir los datos. El periodo de reloj T1 proporciona la dirección de la memoria y las señales de control y el periodo del reloj T2 es donde los datos se transfieren entre la memoria y el microprocesador. Observe que el $\overline{RDY}$ se debe convertir en un cero lógico para ocasionar que la

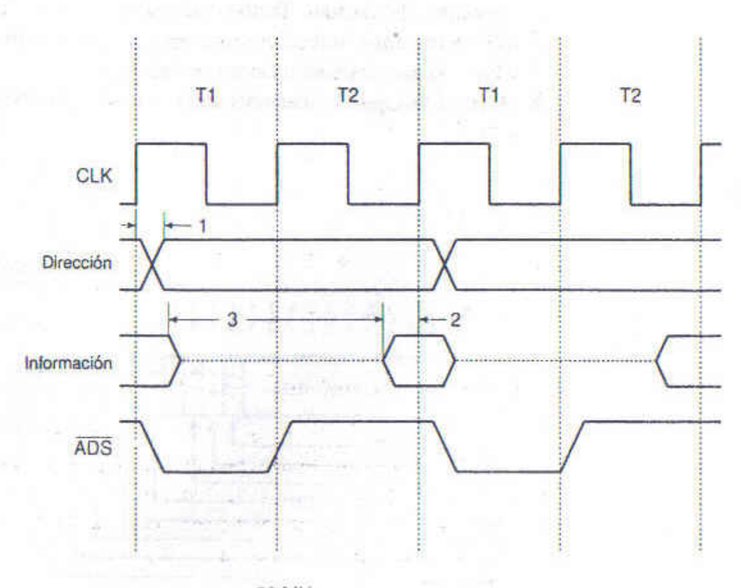
31							24	23							16	15							8	7							0
PG	CE	WT											AM		WP											NE		TS	EM	MP	PE

FIGURA 14–37 Registro de control cero (CR0) para el microprocesador 80486.

información se transfiera y para terminar el ciclo del canal. El tiempo de acceso para un acceso no ráfaga se determinará tomando 2 periodos del reloj menos el tiempo requerido para que la dirección aparezca en el canal de direcciones, menos el tiempo de estabilización para las conexiones del canal de datos. Para la versión de 20 MHz del 80486, dos periodos del reloj requieren de 100 ns menos 28 ns para el tiempo de estabilización de la dirección y 3 ns para el tiempo de estabilización de los datos. Esto proporciona un tiempo de acceso no ráfaga de 100 ns - 31 ns o 69 ns. Por supuesto que el tiempo del decodificador y los tiempos de retardo están incluidos, el tiempo de acceso permitido para la memoria es aún menor para una operación sin estados de espera. Además, si una versión de frecuencia más alta del 80486 se usa en un sistema, el tiempo de acceso a la memoria es aún menor.

La figura 14-39 muestra el diagrama de temporización para llenar una línea caché con cuatro números de 32 bits usando una ráfaga. Observe que las direcciones (A31-A4) aparecen durante T1 y permanecen constantes durante el ciclo de ráfaga. Observe también que A2 y A3 cambian

FIGURA 14–38 Temporización de la lectura por ráfaga para el microprocesador 80486.



	20 MHz	
Tiempo 1:	3 – 28 ns	Tiempo de retardo del direccionamiento
Tiempo 2:	6 ns	Tiempo de estabilización de los datos
Tiempo 3:	76 ns	Tiempo de acceso

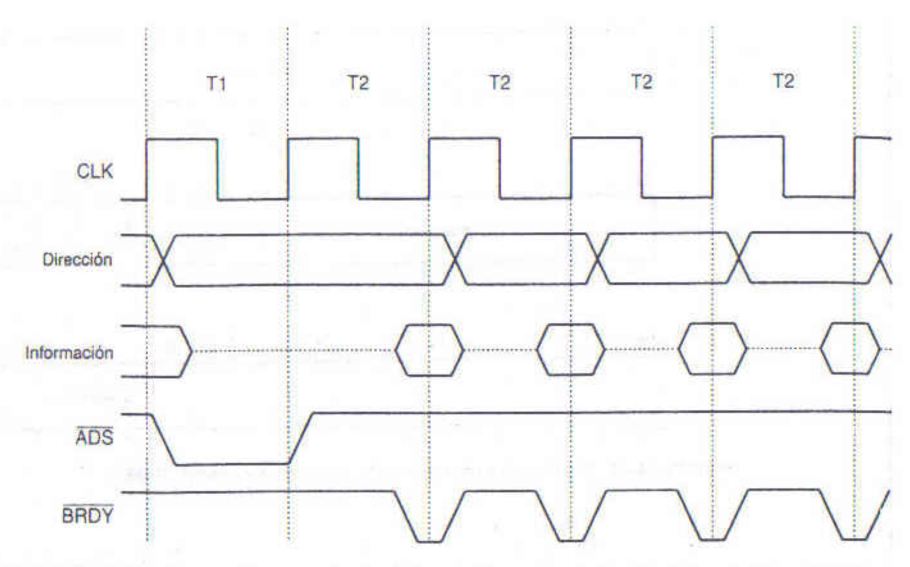


FIGURA 14–39 Un ciclo por ráfaga que lee 4 palabras dobles en 5 periodos de reloj en el sistema 80486.

durante cada T2, después del primero para direccionar cuatro números consecutivos de 32 bits en el sistema de la memoria. Como se menciona, llenar un caché utilizando una ráfaga requiere de sólo cinco periodos de reloj (un T1 y 4 T2) para llenar una línea caché con cuatro dobles palabras de datos. El tiempo de acceso usando una versión de 20 MHz del 80486 para la segunda y subsecuentes dobles palabras es 50 ns - 28 ns - 3 ns, o 19 ns asumiendo que no hay retardos en el sistema. Para usar las transferencias del modo ráfaga, necesitamos memoria de alta velocidad. Debido a que el mejor tiempo de acceso de la memoria DRAM es de 55 ns, estamos obligados a utilizar SRAM para las transferencias del ciclo en ráfaga. Aun con SRAM estamos tocando los límites de la tecnología aun no un microprocesador 80486 de 20 MHz. Observe que la terminal $\overline{BRDY}$ reconoce una transferencia en ráfaga mientras que la terminal $\overline{RDY}$ reconoce una transferencia de memoria normal.

Administrador de memoria del 80486

El 80486 contiene el mismo sistema de administración de la memoria que el 80386. Esto incluye una unidad de paginación para permitir que cualquier bloque de 4K bytes de memoria física sea asignado a cualquier bloque de 4K bytes de memoria lineal. Los tipos de descriptores son exactamente los mismos que para el microprocesador 80386. En realidad, la única diferencia entre el sistema administrador de memoria del 80386 y el sistema administrador de memoria del 80486 es la paginación.

El sistema de paginación del 80486 puede deshabilitar el uso de memoria caché para secciones de páginas de memoria transformadas, mientras que el 80386 no puede hacerlo. La figura 14-40 muestra la tabla de entradas al directorio de las tablas de páginas y la entrada a la tabla de páginas. Si éstos se comparan con los elementos del 80386 en la figura 14-29, la adición de 2 bits

FIGURA 14–40 El directorio de las paginas o elemento de tabla de las páginas para el microprocesador 80486.

31–12	11 10 9	8	7	6	5	4	3	2	1	0
Tabla de páginas o Marco de página	OS Bits	O	O	D	A	PCD	PWT	US	RW	P

de control nuevos se observa (PWT y PCD). La página de escritura a través (PWT) y el control de deshabilitación de página de caché (PCD) manejan el uso del caché.

El PWT controla cómo funciona el caché para una operación de escritura a la memoria caché externa. No controla la escritura en el caché interno. El nivel lógico de este bit se encuentra en la terminal PWT del microprocesador 80486. Por lo tanto, externamente se puede dictar la política de escritura a través del caché externo.

El bit PCD controla el caché integrado. Si el PCD = 0, el caché integrado está habilitado para la página actual de la memoria. Observe que los elementos de la tabla de páginas del 80386 colocan un 0 lógico en la posición del bit PCD habilitando el uso caché. Si PCD = 1, se deshabilita al caché integrado. El uso caché está deshabilitado independientemente de la condición de $\overline{KEN}$, CD y NW.

14–9 CONJUNTO DE INSTRUCCIONES DEL 80486

El conjunto de instrucciones del 80486 es casi idéntico al conjunto de instrucciones del 80386, excepto por unas cuantas instrucciones adicionales. La diferencia principal es que el 80486 ejecuta muchas instrucciones en un periodo del reloj mientras que el 80386 requiere de 2. Este incremento es responsable de una mejoría en la velocidad general de aproximadamente el 50 por ciento para el 80486 cuando se compara con el microprocesador 80386. La tabla 14-8 muestra las instrucciones nuevas para el microprocesador 80486. Observe que ya que el 80486 también incluye un coprocesador numérico 80387 integrado, las instrucciones del coprocesador mencionadas en el Capítulo 12 también se aplican al conjunto de instrucciones del 80486.

La instrucción BSWAP permite que sean cambiados bytes en cualquier registro extendido de 32 bits. Cuando la instrucción BSWAP se ejecuta, intercambia el byte más a la derecha con el byte más a la izquierda y también intercambia a los dos bytes del centro. Por ejemplo, si EAX = 01234567H y la instrucción BSWAP EAX se ejecuta, el resultado es 67452301H.

CMPXCHG compara el operando destino con el contenido del acumulador (EAX, AX y AL). Si el operando destino es igual al acumulador, el operando fuente será copiado al operando destino. Si el operando destino no es igual al acumulador, el operando destino se copia al acumulador. Un ejemplo es un CMPXCHG CX,DX, que copia a DX a CX si DX = AX; de otra manera copia CX a AX. En todos los casos la instrucción CMPXCHG cambia los bits de bandera para indicar el resultado de la comparación. Otro ejemplo es CMPXCHG CL,DATO, el cual copia el contenido de la localidad de la memoria DATO a CL si CL = AL; de otra manera copia a CL AL.

La instrucción INVD (invalidar caché de datos) vacía el contenido del caché de datos actual sin escribir los cambios al sistema de la memoria. Debe tener cuidado cuando use esta instrucción porque se puede perder información si INVD es ejecutado antes de que los datos se escriban a

TABLA 14-8 Instrucciones nuevas para el microprocesador 80486

Instrucción	*Función*
BSWAP	Cambia los bytes en un registro
CMPXCHG	Compara e intercambia
INVD	Borra la memoria caché integrada
INVLPG	Invalida la entrada a TLB
WBINVD	Borra la memoria caché interna después de escribir líneas sucias a la memoria
XADD	Intercambiar y sumar

la memoria del sistema desde el caché. Esta instrucción es principalmente usada a nivel del sistema.

La instrucción INVLPG (invalidar la entrada de la TLB) invalida la entrada a un elemento en el drear de transformaciones frecuentes (TLB) usado por el sistema de solicitud de páginas en un sistema de memoria virtual. La instrucción calcula la dirección del operando y la saca de la TLB si la entrada se mapeó a la TLB. Como con la instrucción INVD, esta instrucción será usada al nivel del sistema operativo en vez del nivel de aplicación.

La instrucción WBINVD (escribir antes de invalidar caché de información) funciona como la instrucción INVD, excepto que el contenido de cualquier ubicación en el caché de datos es escrito primero a la memoria antes de que el caché sea borrado. Esta instrucción normalmente se encuentra a nivel del sistema.

La instrucción XADD intercambia y suma los datos en dos registros o entre la memoria y un registro. Por ejemplo, la instrucción XADD EAX,EBX suma a EBX a EAX y almacena la suma en EAX, tal como una instrucción de suma. La diferencia es que el valor original en EAX se transfiere a EBX. Como la suma afecta a las banderas, también lo hace la instrucción XADD.

Registros para prueba caché

Aunque no son instrucciones, los registros de prueba del caché son colocados en esta sección para mostrar el uso de los registros de prueba caché y algo de la programación para el microprocesador 80486. Los registros para prueba del caché del 80486 son TR3 (registro de datos de caché), TR4 (registro para prueba de estado del caché) y TR5 (registro para prueba del control del caché) que no están definidos para el microprocesador 80386. Estos tres registros están mostrados en la figura 14-41.

El registro de datos de caché (TR3) es usado para tener acceso ya sea al área de llenado de el caché para una operación de prueba de escritura o al área de lectura del caché para una operación de prueba de lectura caché. Este registro es una ventana en la memoria caché de 8K bytes integrada del 80486 y se utiliza para probar el caché. Para llenar o leer una línea caché (128 bits de ancho), TR3 se debe escribir o leer cuatro veces.

El contenido del campo de selección en TR5 determina cuál línea del caché interno se escribe o lee por medio de TR3. El campo de prueba de 7 bits selecciona una de las 128 diferentes líneas del caché de 16 bytes de ancho. Los bits de selección de entrada de TR5 seleccionan una entrada en el conjunto o la localidad de 32 bits en el área para llenar/leer. Los bits de control de TR5

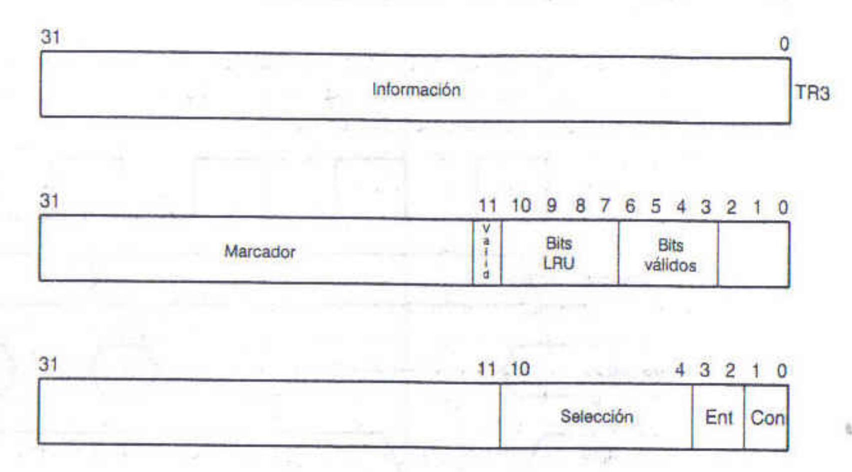


FIGURA 14-41 Registro para pruebas caché del microprocesador 80486.

habilitan a la operación (00), de llenado o lectura del área, realizar una escritura en caché (01), una lectura en caché (10) o borrar el caché (11).

El registro para estado del caché (TR4) contiene el marcador del caché, los bits LRU y un bit válido. Este registro está cargado con el marcador y bit válido antes de una operación de escritura en caché y contiene el marcador, un bit de validez, bits LRU y cuatro bits de validez en una lectura de prueba en caché.

El caché se prueba cada vez que el microprocesador se reinicializa si la terminal AHOLD está en alto durante 2 ciclos de reloj antes de que la terminal RESET vaya abajo. Esto causa que el 80486 se pruebe a sí mismo totalmente con una autocomparación integrada o BIST. La BIST utiliza TR3, TR4 y TR5 para probar completamente el caché integrado. El resultado de la prueba se reporta en el registro EAX. Si EAX es cero, el microprocesador, el coprocesador y el caché han pasado la auto-comparación. El valor de EAX puede ser probado después de una reinicialización para determinar cuándo se ha detectado un error. En la mayoría de los casos no tenemos acceso directo a los registros para prueba a menos que deseemos realizar nuestras propias pruebas en el caché o TLB.

14-10 RESUMEN

1. El microprocesador 80386 en una versión mejorada del microprocesador 80286 que incluye una unidad de administración de memoria mejorada para proporcionar la paginación de la memoria. El 80386 también incluye los registros extendidos de 32 bits y un canal de direcciones y datos de 32 bits. Una versión reducida del 80386DX con un canal de datos de 16 bits y de direcciones de 24 bits está disponible como el microprocesador 80386SX.

2. El 80386 tiene un tamaño físico de la memoria de 4G bytes que puede ser accesado como una memoria virtual hasta 64T bytes. La memoria del 80386 es de 32 bits de ancho y está accesada como bytes, palabras o dobles palabras.
3. Cuando el 80386 se opera en el modo de paralelismo, envía la dirección de la próxima instrucción o dato de la memoria al sistema de la memoria antes de terminar la ejecución de la instrucción actual. Esto permite que el sistema de la memoria comience a la próxima instrucción o dato antes de que la actual esté terminada. Esto incrementa el tiempo de acceso, por lo tanto reduciendo la velocidad de la memoria.
4. Un sistema de memoria caché permite que los datos que se leen frecuentemente sean accesados en menos tiempo, porque están almacenados en una memoria semiconductora de alta velocidad. Si los datos se escriben a la memoria, también se escriben al caché, así que la información más actual siempre está en el caché.
5. La estructura E/S del 80386 es casi idéntica a la del 80286, excepto que la E/S puede inhibir cuando el 80386 se opere en el modo protegido por medio del mapa de bits de protección E/S almacenado en el TSS.
6. El conjunto de registros del 80386 contiene versiones extendidas de los registros introducidos en el microprocesador 80286. Estos registros extendidos incluyen: EAX, EBX, ECX, EDX, EBP, ESP, EDI, ESI, EIP y EFLAGS. Además de los registros extendidos, se agregan dos registros de segmento suplementarios (FS y GS). Los registros para depuración y control manejan las tareas de depuración en el sistema y la administración de la memoria en el modo protegido.
7. El conjunto de instrucciones del 80386 está mejorado para incluir instrucciones que accesan al conjunto de registros extendidos de 32 bits. Las mejorías también incluyen modos adicionales de direccionamiento que permiten que cualquier registro extendido direccione datos de la memoria. Fue agregado escalamiento para que un registro índice se pueda multiplicar por 1, 2, 4 u 8. Los nuevos tipos de instrucciones incluyen rastreo de bits, transferencia en cadenas con signos o extensión de cero, inicialización de bytes sobre condición y corrimientos de doble precisión.
8. Las interrupciones, en el microprocesador 80386, han sido expandidas para que incluyan interrupciones predefinidas adicionales en la tabla de vectores de interrupción. Estas interrupciones adicionales son usadas en el sistema administrador de memoria.
9. El administrador de memoria del 80386 es parecido al del 80286, excepto que las direcciones físicas generadas por el MMU son de 32 bits de ancho en vez de 24 bits. El MMU del 80386 también tiene la capacidad de paginación.
10. El 80386 es operado en el modo real (modo 8086) cuando se reinicializa. El modo real permite que el microprocesador accese información en el primer 1M byte de memoria. En el modo protegido, el 80386 accesa cualquier localidad en su intervalo de direcciones físicas de 4G bytes.
11. Un descriptor es una serie de 8 bytes que especifica cómo un segmento de código o datos es utilizado por el 80386. El descriptor es seleccionado por un selector que está almacenado en uno de los registros de segmento. Los descriptores son usados sólo en el modo protegido.
12. La administración de la memoria se logra por medio de una serie de descriptores almacenados en las tablas de descriptores. Para facilitar la administración de la memoria, el 80386 usa tres tablas de descriptores: la tabla de descriptores globales (GDT), la tabla de descriptores locales (LDT) y la tabla de descriptores de interrupciones (IDT). La GDT y LDT contienen cada una hasta 8,192 descriptores, mientras que la IDT contiene hasta 256 descriptores. La

GDT y la LDT describen los segmentos de código y datos y también las tareas. El IDT describe los 256 diferentes niveles de interrupción por medio de descriptores de las compuertas para interrupciones.

13. El TSS (segmento del estado de las tareas) contiene información sobre la tarea actual y también sobre la tarea anterior. Añadido al final del TSS está un mapa de protección de bits E/S que inhibe las direcciones de puertos de E/S seleccionadas.
14. El mecanismo de paginación de la memoria permite que cualquier página de la memoria física de 4K bytes sea distribuida a cualquier página de la memoria lineal de 4K bytes. Por ejemplo, la localidad de la memoria 00A00000H puede ser asignada a la localidad de la memoria A0000000H por medio del mecanismo de paginación. Un directorio de páginas y tablas de páginas se usan para asignar cualquier dirección física a cualquier dirección lineal. El mecanismo de paginación se puede utilizar en el modo protegido o en el modo virtual.
15. El microprocesador 80486 es una versión mejorada del microprocesador 80386 que tiene integrados un caché de 8K bytes, un coprocesador aritmético 80387 y ejecuta muchas instrucciones en un periodo de reloj.
16. El microprocesador 80486 ejecuta unas cuantas instrucciones nuevas que controlan la memoria caché interna y permiten sumar (XADD) y comparar (CMPXCHG) con una operación de intercambio y cambio de bytes (BSWAP). Además de estas cuantas instrucciones adicionales, el 80486 es cien por ciento compatible hacia arriba con el 80386 y 80387.
17. Una nueva característica encontrada en el 80486 es el BIST (auto-comprobación integrada) que prueba el microprocesador, coprocesador y caché en la reinicialización. Si el 80486 pasa la prueba, EAX contiene un cero.
18. Registros de pruebas adicionales se agregaron al 80486 para permitir que la memoria caché se pruebe. Estos nuevos registros para prueba son TR3 (información del caché), TR4 (estado del caché) y TR5 (control del caché). Aunque rara vez usamos estos registros, son utilizados por BIST cada vez que BIST se ejecuta después de una operación de reinicialización.

14-11 PREGUNTAS Y PROBLEMAS

1. El microprocesador 80386 accesa __________ bytes de memoria física cuando se opera en el modo protegido.
2. El microprocesador 80386 accesa ______________ bytes de memoria virtual por medio de su unidad de administración de memoria.
3. Describa las diferencias entre el 80386DX y el 80386SX.
4. Dibuje el mapa de la memoria del 80386 cuando se opera en el
   (a) modo protegido
   (b) modo real
5. ¿Cuánta corriente está disponible en las conexiones de salida de datos del 80386? Compare estas corrientes con las corrientes disponibles en la misma conexión de salida de un microprocesador 8086.
6. Describa el sistema de memoria del 80386 y explique el propósito y funcionamiento de las señales para selección de banco.

7. Explique la acción de una reinicialización dura (hardware) en las conexiones del canal de direcciones del 80386.
8. Explique cómo la arquitectura paralela alarga el tiempo de acceso para muchas referencias de la memoria en un sistema basado en el microprocesador 80386.
9. Describa brevemente cómo funciona el sistema de memoria caché.
10. Los puertos de E/S en el 80386 comienzan en la dirección ____________ y se extienden a la dirección E/S ________.
11. ¿Qué puertos E/S comunican información entre el 80386 y el coprocesador 80387?
12. Compare y contraste las conexiones de memoria y de E/S del 80386 con las del 8086.
13. ¿Si el 80386 funciona a 20 MHz, qué frecuencia del reloj será aplicada a la terminal CLK2?
14. ¿Cuál es el propósito de la terminal $\overline{BS16}$ en el microprocesador 80386?
15. ¿Cuáles dos registros adicionales de segmento se encuentran en el modelo de programación para el 80386 que no están presentes en el 8086?
16. Mencione los registros extendidos encontrados en el microprocesador 80386.
17. Mencione cada bit del registro de banderas del 80386 y describa su propósito.
18. Defina el propósito de cada uno de los registros de control (CR0, CR1, CR2 y CR3) del 80386.
19. Defina el propósito de cada uno de los registros de depuración del 80386.
20. ¿Los registros para depuración causan qué nivel de interrupción?
21. Describa el funcionamiento de la instrucción de rastreo de bit hacia adelante.
22. Describa el funcionamiento de la instrucción de rastreo de bit hacia atrás.
23. Describa el funcionamiento de la instrucción SHRD.
24. Forme una instrucción que accese los datos del segmento FS en la localidad direccionada indirectamente por el registro DI. La instrucción debe almacenar el contenido de EAX a esta localidad de la memoria.
25. ¿Qué es un direccionamiento por índice escalado?
26. ¿Es legal la siguiente instrucción? MOV AX,[EBX+ECX].
27. Explique cómo las siguientes instrucciones calculan la dirección de la memoria:
    (a) ADD[EBX+8*ECX],AL
    (b) MOV DATO[EAX+EBX],CX
    (c) SUB EAX,DATO
    (d) MOV ECX,[EBX]
28. ¿Cuál es el propósito de la interrupción 7?
29. ¿Qué número de vector interrupción se activa para una violación de privilegio de protección?
30. ¿Qué es una falla de doble interrupción?
31. Cuando una interrupción ocurre en el modo protegido, ¿qué define a los vectores de interrupciones?
32. ¿Qué es un descriptor?
33. ¿Qué es un selector?
34. ¿Cómo elige el selector a la tabla de descriptores locales?
35. ¿Qué registro es usado para direccionar a la tabla de descriptores globales?
36. ¿Cuántos descriptores globales pueden ser almacenados en la GDT?
37. Explique cómo el 80386 puede direccionar un espacio de la memoria virtual de 64T bytes cuando la memoria física contiene sólo 4G bytes de memoria.
38. ¿Cuál es la diferencia entre un descriptor de segmento y un descriptor de sistema?
39. ¿Qué es el segmento del estado de las tareas (TSS)?

40. ¿Cómo se accesa TSS?
41. Describa cómo el 80386 cambia del modo real al modo protegido.
42. Describa cómo el 80386 cambia del modo protegido al modo real.
43. ¿Cuál es el modo de operación virtual 8086 del microprocesador 80386?
44. ¿Cómo localiza el directorio de páginas el 80386?
45. ¿Cuántos bytes se encuentran en una página de la memoria?
46. Explique cómo la dirección lineal de memoria D0000000H se puede asignar a la dirección de memoria física C0000000H con la unidad de paginación del 80386.
47. ¿Cuáles son las diferencias entre el microprocesador 80386 y 80486?
48. ¿Cuál es el propósito de la terminal de entrada $\overline{\text{FLUSH}}$ en el microprocesador 80486?
49. Compare el conjunto del registro del 80386 con el del microprocesador 80486.
50. ¿Qué diferencias existen en las banderas del 80486 cuando son comparadas al microprocesador 80386?
51. ¿Qué terminales son utilizadas para la verificación de paridad en el microprocesador 80486?
52. El microprocesador 80486 utiliza _______________ paridad.
53. El caché integrado del microprocesador 80486 es de ______ K bytes.
54. Una línea caché se llena leyendo _______ bytes del sistema de la memoria.
55. ¿Qué es una ráfaga en el 80486?
56. Defina el término escritura a través del caché.
57. ¿Qué es BIST?
58. ¿Puede ser deshabilitado el caché del 80486 por programa? (Explique su respuesta.)
59. Explique cómo es que la instrucción XADD EBX,EDX funciona.
60. La instrucción CMPXCHG CL,AL compara a CL con AL. ¿Qué más ocurre cuando esta instrucción se ejecuta?
61. Compare la instrucción INVD con la instrucción WBINVD.
62. ¿Cuál es el propósito del bit PCD en el directorio de la tabla de la página o entrada a la tabla de página?
63. ¿Afecta el bit PWT al directorio de la tabla de páginas o entrada de la tabla de página el caché integrado?

APENDICE A

El ensamblador y el sistema operativo DOS

Este apéndice fue proporcionado para que el uso del ensamblador se pueda entender y también para mostrar las solicitudes a funciones del DOS (sistema operativo del disco) y BIOS (sistema básico de E/S) que son utilizados por el lenguaje de ensamblador para controlar a la IBM-PC o a su copia. Las solicitudes a funciones controlan todo desde leer y escribir información al disco hasta manejar el teclado y las pantallas. El ensamblador mostrado en este texto son los programas macros de ensamblador de Microsoft ML (Versión 6.0) y MASM (versión 5.10).

USO DEL ENSAMBLADOR

El programa del ensamblador requiere que primero sea escrito un programa simbólico, usando un procesador de palabra, editor de texto, o el programa workbench proporcionado con el paquete del ensamblador. El editor proporcionado con la versión 5.10 es M.EXE y es estrictamente un editor de pantalla completa. El editor proporcionado con la versión 6.0 es PWB.EXE y es un sistema de desarrollo completamente integrado que contiene una ayuda extensa. Refiérase a la documentación que acompaña a su paquete del ensamblador para más detalles sobre el funcionamiento del programa editor. Si es posible utilice la versión 6.0 del ensamblador porque contiene un archivo de ayuda detallado que guía al usuario por las instrucciones del lenguaje de ensamblador, las directivas, y hasta las solicitudes a las funciones de los interruptores del DOS y BIOS.

Si está usando un procesador para desarrollar su software, asegúrese que esté configurado para generar un archivo ASCII puro. El archivo fuente que usted genere debe utilizar la extensión .ASM que será requerida por el ensamblador para identificar correctamente su programa fuente.

Una vez que su archivo fuente está preparado, debe ser ensamblado. Cuando está utilizando el workbench proporcionado con la versión 6.0, lo logra seleccionando la característica de compilación con su mouse. Si está usando un procesador de palabra y las líneas de comando de DOS con la versión 5.10, entonces vea el Ejemplo A-1 para el diálogo de la versión 5.10 para ensamblar un archivo llamado FROG.ASM. Observe que este ejemplo muestra en itálicas las partes escritas por el usuario.

Una vez que un programa sea ensamblado, debe ser formado antes de que se pueda ejecutar. El compilador convierte al archivo objeto en un archivo ejecutable (.EXE). El Ejemplo A-2 muestra

el diálogo requerido para el compilador usando un archivo objeto MASM versión 5.10. Cuando el ensamblador ML versión 6.0 se está usando, éste automáticamente ensambla y forma un programa utilizando el comando COMPILE o BUILD de workbench. Después de compilar con ML, workbench permite que el programa sea revisado con una herramienta para verificar llamado *code view*. El *code view* también está disponible con MASM, pero se debe escribir CV en la línea de comando del DOS para tener acceso a él.

EJEMPLO A–1

```
A>MASM

Microsoft (R) Macro Ensamblador Versión 5.10
Copyright (C) Microsoft Corp 1981, 1989. Todos los derechos reservados.

Nombre del archivo fuente [.ASM] :FILE
Nombre del archivo objeto [FILE.OBJ] :FILE
Nombre del archivo fuente [NUL.LST] :FILE
Referencia [NUL.CRF] :FILE
```

EJEMPLO A–2

```
A>LINK

Microsoft (R) Overlay Linker Versión 3.64
Copyright (C) Microsoft Corp. 1983-1988. Todos los derechos reservados.

Módulos para Objetos [.OBJ] :TEST
Correr Archivo [TEST.EXE] :TEST
Mencionar archivo [NUL.MAP] :TEST
Bibliotecas [.LIB] :SUBR
```

La versión 6.0 del programa MASM de Microsoft contiene el programa workbench del programador. Este programa permite que un programa en el lenguaje de ensamblador sea desarrollado con un editor de pantalla completa y una barra de herramientas. La Figura A-1 muestra la pantalla encontrada en el workbench del programador. Para tener acceso a este programa escriba PWB en el símbolo de comando DOS. La opción de MAKE permite que un programa sea automáticamente ensamblado y formado, haciendo que estas tareas sean sencillas en comparación a la versión 5.10 del ensamblador.

MODELOS DE LA MEMORIA DEL ENSAMBLADOR

Los modelos de la memoria del comando .MODEL fueron introducidos en el capítulo 6. Aquí definimos completamente a los modelos de la memoria disponibles para el desarrollo de software. Cada modelo define la manera en que un programa es almacenado en el sistema de la memoria. La Tabla A-1 describe los diferentes modelos disponibles con MASM y también ML.

Observe que el modelo miniatura se utiliza para crear un archivo .COM en vez de un archivo ejecutable. El archivo .COM es diferente porque toda la información y los códigos entran en un segmento del código. Un archivo .COM debe tener el programa originado para que comience en la dirección desplazada de 0100H. Un archivo .COM se carga del disco y se ejecuta más rápido

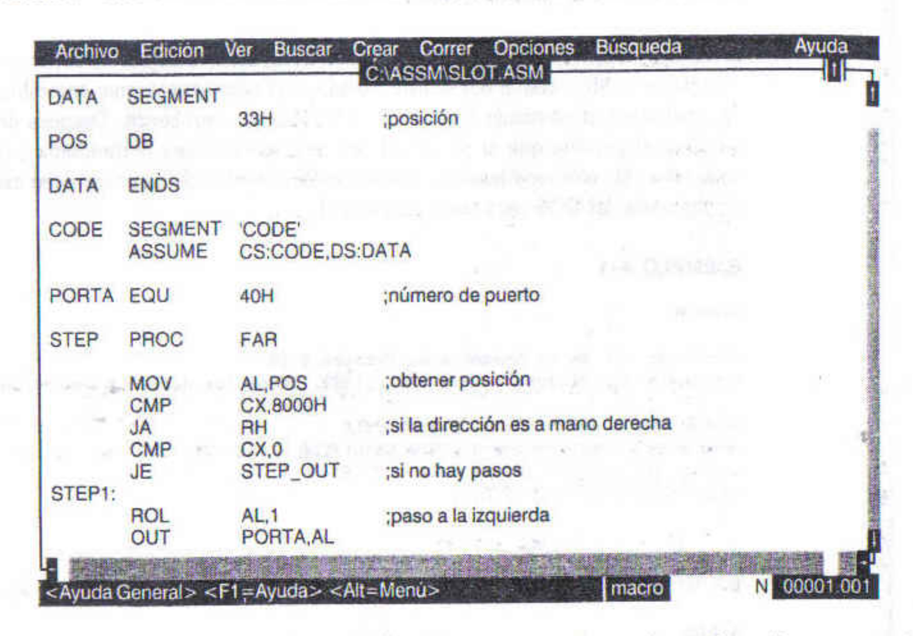


FIGURA A–1 La pantalla para edición del workbench de un programador usada para desarrollar programas de lenguaje de ensamblaje.

TABLA A–1 Modelos de la memoria para el ensamblador

Tipo	*Descripción*
Miniatura	Toda la información y código caben en un segmento, el segmento para código. Los programas de modelo miniatura son escritos en el formato del archivo .COM lo cual significa que el programa debe originarse en la localidad 0100H de la memoria. Este modelo es más usado con programas pequeños.
Pequeño	Toda la información cabe en un solo segmento de información de 64K byte y todo el código cabe en otro segmento de 64K byte. Esto permite que se tenga acceso a todo el código con casi cambios y solicitudes.
Mediano	Toda la información cabe en un solo segmento de información de 64K byte, y el código cabe en más de un segmento de código. Esto permite que el código exista en múltiples segmentos.
Compacto	Todo el código cabe en un solo segmento de 64K byte, y la información cabe en más de un segmento de información.
Grande	El código y la información caben en múltiples segmentos de código e información.
Gigante	Igual que el grande, pero permite segmentos de información que son más grandes que 64K.
Plano	No está disponible en MASM versión 5.10. El modelo plano de la memoria utiliza un segmento con una distancia máxima de 512M bytes para almacenar información y código.

que el archivo ejecutable normal (.EXE). Para la mayoría de las aplicaciones normalmente usamos el archivo ejecutable (.EXE) y el modelo de la memoria pequeño.

Cuando los modelos son usados para crear un programa, ciertos valores predeterminados se aplican como es mostrado en la Tabla A-2. El directivo en esta tabla es utilizado para arrancar un tipo de segmento específico para los modelos listados en la tabla. Cuando el directivo .CODE es colocado en un programa, indica el comienzo del segmento de código. Asimismo, .DATA indica el comienzo de un segmento de información. La columna de nombre indica el nombre del segmento. Alinear indica cuando el segmento está alineado en una palabra, doble palabra, o un párrafo de 16-bytes. Combinar indica el tipo de segmento creado. La clase indica la clase del segmento, tal como 'CODE' o 'DATA'. El grupo indica el tipo de grupo del segmento.

El Ejemplo A-3 muestra un programa que utiliza el modelo pequeño. El modelo pequeño se usa para programas que contienen un segmento de DATA y uno de CODE. Esto se aplicará a muchos programas que son desarrollados. Observe que no sólo está mostrado un programa, pero también es toda la información generada por el ensamblador. Aquí el directivo .DATA y el directivo .CODE indican el comienzo de los segmentos. También observe cómo el registro DS se carga en este programa.

EJEMPLO A–3 (página 1 de 2)

```
Microsoft (R) Macro Ensamblador Versión 6.00

                           .MODEL SMALL
                           .STACK 100H
0000                       .DATA

0000 0A                    FROG      DB       10
0001 0064 [                DATA1     DB       100 DUP (2)
              02
                ]

0000                       .CODE

0000 B8 ---- R             BEGIN:   MOV      AX,DGROUP              ;establecer DS
0003 8E D8                          MOV      DS,AX
                                    .
                                    .
                                    .
                                    END      BEGIN
Segments and Groups:

        N a m e            Size Length  Align Combine Class

DGROUP .............. GROUP
_DATA ............... 16 Bit      0065      Word      Public 'DATA'
STACK ............... 16 Bit      0100      Para       Stack 'STACK'
_TEXT ............... 16 Bit      0005      Word      Public 'CODE'

Symbols:

        N a m e          Type     Value    Attr

@CodeSize ........... Number   0000h
@DataSize ........... Number   0000h
@Interface .......... Number   0000h
@Model .............. Number   0002h
```

TABLA A–2 Valores predeterminados para el directivo .MODEL

Modelo	*Directivo*	*Nombre*	*Alineación*	*Combinación*	*Clase*	*Grupo*
Miniatura	.CODE	_TEXT	Word	PUBLIC	'CODE'	DGROUP
	.FARDATA	FAR_DATA	Para	Private	'FAR_DATA'	
	.FARDATA?	FAR_BSS	Para	Private	'FAR_BSS'	
	.DATA	_DATA	Word	PUBLIC	'DATA'	DGROUP
	.CONST	CONST	Word	PUBLIC	'CONST'	DGROUP
	.DATA?	_BSS	Word	PUBLIC	'BSS'	DGROUP
Pequeño	.CODE	_TEXT	Word	PUBLIC	'CODE'	
	.FARDATA	FAR_DATA	Para	Private	'FAR_DATA'	
	.FARDATA?	FAR_BSS	Para	Private	'FAR_BSS'	
	.DATA	_DATA	Word	PUBLIC	'DATA'	DGROUP
	.CONST	CONST	Word	PUBLIC	'CONST'	DGROUP
	.DATA?	_BSS	Word	PUBLIC	'BSS'	DGROUP
	.STACK	STACK	Para	STACK	'STACK'	DGROUP
Mediano	.CODE	name_TEXT	Word	PUBLIC	'CODE'	
	.FARDATA	FAR_DATA	Para	Private	'FAR_DATA'	
	.FARDATA?	FAR_BSS	Para	Private	'FAR_BSS'	
	.DATA	_DATA	Word	PUBLIC	'DATA'	DGROUP
	.CONST	CONST	Word	PUBLIC	'CONST'	DGROUP
	.DATA?	_BSS	Word	PUBLIC	'BSS'	DGROUP
	.STACK	STACK	Para	STACK	'STACK'	DGROUP
Compacto	.CODE	_TEXT	Word	PUBLIC	'CODE'	
	.FARDATA	FAR_DATA	Para	Private	'FAR_DATA'	
	.FARDATA?	FAR_BSS	Para	Private	'FAR_BSS'	
	.DATA	_DATA	Word	PUBLIC	'DATA'	DGROUP
	.CONST	CONST	Word	PUBLIC	'CONST'	DGROUP
	.DATA?	_BSS	Word	PUBLIC	'BSS'	DGROUP
	.STACK	STACK	Para	STACK	'STACK'	DGROUP
Grande o gigante	.CODE	name_TEXT	Word	PUBLIC	'CODE'	
	.FARDATA	FAR_DATA	Para	Private	'FAR_DATA'	
	.FARDATA?	FAR_BSS	Para	Private	'FAR_BSS'	
	.DATA	_DATA	Word	PUBLIC	'DATA'	DGROUP
	.CONST	CONST	Word	PUBLIC	'CONST'	DGROUP
	.DATA?	_BSS	Word	PUBLIC	'BSS'	DGROUP
	.STACK	STACK	Para	STACK	'STACK'	DGROUP
Plano	.CODE	_TEXT	Dword	PUBLIC	'CODE'	
	.FARDATA	_DATA	Dword	PUBLIC	'DATA'	
	.FARDATA?	_BSS	Dword	PUBLIC	'BSS'	
	.DATA	_DATA	Dword	PUBLIC	'DATA'	
	.CONST	CONST	Dword	PUBLIC	'CONST'	
	.DATA?	_BSS	Dword	PUBLIC	'BSS'	
	.STACK	STACK	Dword	PUBLIC	'STACK	

EJEMPLO A–3 (página 2 de 2)

```
@code ............. Text               _TEXT
@data ............. Text               DGROUP
@fardata? ......... Text               FAR_BSS
@fardata .......... Text               FAR_DATA
@stack ............ Text               DGROUP
BEGIN ............. L Near  0000       _TEXT
DATA1 ............. Byte    0001       _DATA
FROG .............. Byte    0000       _DATA

        0 Warnings
        0 Errors
```

El ejemplo A-4 muestra un programa que utiliza el modelo grande. Observe cómo difiere del programa del modelo pequeño en el Ejemplo A-3. Los modelos pueden ser muy útiles para desarrollar software, pero frecuentemente utilizamos descripciones para segmento completo como fue descrito en el Capítulo 6.

EJEMPLO A–4 (página 1 de 2)

```
Microsoft (R) Macro Ensamblador Versión 6.00                07/30/91 22:38:58

                          .MODEL LARGE
                          .STACK 1000H
0000                      .FARDATA?

0000 00                   FROG    DB     ?
0001 0064 [               DATA1   DW     100 DUP (?)
          0000
               ]

0000                      .CONST

0000 54 68 69 73 20 69 MES1    DB     ;'Esta es una cadena de caracteres'
     73 20 61 20 63 68
     61 72 61 63 74 65
     72 20 73 74 72 69
     6E 67
001A 53 6F 20 69 73 20 MES2    DB     ;'También esto lo es!'
     74 68 69 73 21

0000                      .DATA

0000 0000C                DATA2   DW     12
0002 00C8 [               DATA3   DB     200 DUP (1)
          01
             ]

0000                      .CODE

0000                      FUNC    PROC   FAR
                                  .
                                  .
                                  .
0000 CB                           RET

0001                      FUNC    ENDP
```

EJEMPLO A–4 (página 2 de 2)

```
                                END     FUNC

Segments and Groups:

        N a m e         Size        Length   Align   Combine Class

D GROUP............GROUP
_DATA ............ 16 Bit        00CA.    Word               Public 'DATA'
STACK ............ 16 Bit        1000     Para     Stack 'STACK'
CONST ............ 16 Bit        0025     Word               Public 'CONST'     ReadOnly
EXA_TEXT ......... 16 Bit        0001     Word               Public 'CODE'
FAR_BSS .......... 16 Bit        00C9     Para Private 'FAR_BSS'
_TEXT ............ 16 Bit        0000     Word Public 'CODE'

Procedures, parameters and locals:

        N a m e         Type        Value    Attr

FUNC ............. P Far         0000     EXA_TEXT         Length=0001 Public

Symbols:

        N a m e         Type        Value    Attr

@CodeSize ........ Number        0001h
@DataSize ........ Number        0001h
@Interface ....... Number        0000h
@Model ........... Number        0005h
@code ............ Text                   EXA_TEXT
@data ............ Text                   DGROUP
@fardata? ........ Text                   FAR_BSS
@fardata ......... Text                   FAR_DATA
@stack ........... Text                   DGROUP
DATA1 ............ Word          0001     FAR_BSS
DATA2 ............ Word          0000     _DATA
DATA3 ............ Byte          0002     _DATA
FROG ............. Byte          0000     FAR_BSS
MES1 ............. Byte          0000     CONST
MES2 ............. Byte          001A     CONST

             0 Warnings
             0 Errors
```

SOLICITUDES A LAS FUNCIONES DEL DOS

EJEMPLO A–5

```
0000 B4 06                  MOV     AH,6
0002 B2 41                  MOV     DL,'A'
0004 CD 21                  INT     21H
```

Para poder usar las solicitudes a las funciones del DOS, coloque el número de la función en el registro AH y cargue toda la demás información pertinente a los registros como está explicado en la tabla. Una vez que se logra esto, continúe con un INT 21H para ejecutar la función DOS. El Ejemplo A-5 muestra cómo mostrar un ASCII A en la pantalla CTR en la posición actual del cursor con una solicitud de una función del DOS. A continuación está una lista completa de las solicitudes

para funciones del DOS. Observe que algunas de ellas requieren de un segmento y una dirección desplazada, indicado como DS:DI, por ejemplo. Esto significa que el segmento de datos es la dirección del segmento y DI es la dirección desplazada. Todas las solicitudes de una función utilizan INT 21H y AH contiene el número de la solicitud de la función. Observe que las funciones marcadas con @ no se deben utilizar al menos que se esté usando la versión 2.XX de DOS. Como un regla, las solicitudes de una función del DOS guardan todos los registros sin utilizar como la información de salida, pero en ciertos casos algunos registros pueden cambiar. Para poder prevenir problemas, es recomendable guardar registros en donde ocurren problemas.

00H	**TERMINAR UN PROGRAMA**
Entrada	AH = 00H CS = dirección del prefijo para el segmento del programa
Salida	Se teclea DOS
01H	**LEER EL TECLADO**
Entrada	AH = 01H
Salida	AL = caracter ASCII
Notas	Cuando la solicitud de función AL = 00H se debe reutilizar para leer un caracter ASCII extendido. Refiérase al Capítulo 6, Tabla 6-1, para un listado de los códigos del teclado extendido ASCII. Esta solicitud de una función automáticamente hace eco cuando está escrita en la pantalla de video.
02H	**ESCRIBIR AL EQUIPO ESTANDAR DE SALIDA DE INFORMACION**
Entrada	AH = 02H AL = caracter ASCII que se debe mostrar
Notas	Esta solicitud de la función normalmente muestra información en la pantalla de video.
03H	**LEER CARACTER DE COM1**
Entrada	AH = 03H
Salida	AL = caracter ASCII leído del puerto de comunicaciones
Notas	Esta solicitud de función lee información del puerto serial de comunicaciones.
04H	**ESCRIBIR A COM1**
Entrada	AH = 04H DL = caracter que se debe enviar a COM1
Notas	Esta función transmite información por medio del puerto de comunicaciones serial.

05H	**ESCRIBIR A LPT1**
Entrada	AH = 05H DL = carácter ASCII que se debe imprimir
Notas	Imprime DL en la impresora de línea conectada a LPT1.
06H	**DIRIGIR LEER/ESCRIBIR DE LA CONSOLA**
Entrada	AH = 06H DL = OFFH o DL = carácter ASCII
Salida	AL = carácter ASCII
Notas	Cuando DL = OFFH se usa, entonces esta función lee la consola. Cuando DL = caracter ASCII, entonces esta función muestra al carácter ASCII en la pantalla de video de la consola. Cuando un caracter se lee del teclado de la consola, la bandera cero (ZF) indica cuándo un caracter fue tecleado. Una condición cero indica que ninguna tecla está escrita y una condición no cero indica que AL contiene el código ASCII de la tecla o un 00H. Cuando AL = 00H, la función debe ser reusada para leer un carácter ASCII extendido del teclado. Observe que la tecla no manda eco a la pantalla de video.
07H	**DIRIGIR LA ENTRADA DE INFORMACIÓN A LA CONSOLA SIN ECO**
Entrada	AH = 07H
Salida	AL = carácter ASCII
Notas	Funcionan exactamente como el número de función 06H con DL = OFFH, pero no regresan de la función hasta que la tecla sea oprimida.
08H	**LEER LA ENTRADA DE INFORMACION ESTANDAR SIN ECO**
Entrada	AH = 08H
Salida	AL = carácter ASCII
Notas	Funciona como la función 07H, excepto que lee al equipo de entrada de información estándar. El aparato de entrada de información estándar puede ser asignado ya sea como el teclado o el puerto COM. Esta función también responde a un Ctrl- break, en donde las funciones 06H y 07H no lo hacen. Un Ctrl-break ocasiona que se ejecute INT 23H.
09H	**MOSTRAR UNA CADENA DE CARACTERES**
Entrada	AH = 09H DS:DX = dirección de la cadena de caracteres
Notas	La cadena de caracteres debe terminar con un ASCII $ (24H). La cadena de caracteres puede ser de cualquier tamaño y puede contener caracteres de control tal como el regreso de línea (0DH) y la alimentación de línea (0AH).

OAH	**LA ENTRADA DE INFORMACION DEL TECLADO CON BUFER**
Entrada	AH = 0AH DS:DX = dirección del búfer para la entrada de información del teclado
Notas	El primer byte del búfer contiene el tamaño del búfer (hasta 255). El segundo byte esta ocupado con el número de caracteres escritos al regresar. Desde el tercer byte hasta el final del búfer contiene la cadena de caracteres escrita seguida por un retorno (0DH). Esta función continúa leyendo el teclado (mostrando información como fue tecleada) hasta que el número especificado de caracteres sea escrito o hasta que la tecla de retorno (enter) sea tecleada.
OBH	**ESTADO DE LA PRUEBA DEL EQUIPO ESTANDAR PARA LA ENTRADA DE INFORMACION**
Entrada	AH = 0BH
Salida	AL = estado del equipo para la entrada de información
Notas	Esta función prueba al equipo estándar de entrada de información para determinar si hay información disponible. Cuando AL = 00, no hay información disponible. Cuando AL = 0FFH, entonces cuando hay información disponible que debe ser aceptada utilizando el número de la función 08H.
OCH	**BORRAR EL BUFER DEL TECLADO E INVOCAR LA FUNCION DEL TECLADO**
Entrada	AH-OCH AL = 01H, 06H, 07H o 0AH
Salida	véase la salida para las funciones 01H, 06H, 07H o 0AH
Notas	El búfer del teclado retiene las teclas mientras que los programas ejecutan otras tareas. Esta función vacía o borra al búfer y después invoca a la función del teclado ubicado en el registro AL.
ODH	**ELIMINAR A LOS BUFERES DEL DISCO**
Entrada	AH = 0DH
Notas	Borra todos los nombres de archivos almacenados en los búferes del disco. Esta función no cierra los archivos especificados por los búferes del disco, así que se debe tener cuidado con su uso.
OEH	**SELECCIONAR UNIDAD PREDETERMINADA DE DISCO**
Entrada	AH = 0DH DL = el número deseado de la unidad predeterminada de disco
Salida	AL = el número total de lectores de disco existentes en el sistema
Notas	Lector de disco A = 00H, lector de disco B = 01H, lector de disco C = 02H, etc.

OFH	@ABRIR ARCHIVO CON FCB
Entrada	AH = 0FH DS:DX = dirección del bloque para control de un archivo no abierto (FCB)
Salida	AL = 00H cuando el archivo es encontrado AL = 0FFH cuando el archivo no es encontrado
Notas	El bloque de control de archivo (FCB) es usado sólo con el software del DOS antiguo y nunca se debe utilizar con programas nuevos. Los bloques para control de archivo no permiten nombres de rutas como lo hacen los códigos de función de archivo más nuevos presentados más adelante. La Figura A-2 muestra la estructura del FCB. Para abrir un archivo, el archivo debe estar presente en el disco o ser creado con una solicitud de la función 16H.
10H	**@CERRAR EL ARCHIVO CON FCB**
Entrada	AH = 10H DS:DX = dirección del bloque para control del archivo abierto (FCB)
Salida	AL = 00H cuando el archivo está cerrado AL = OFFH cuando es encontrado un error
Notas	Los errores que ocurren normalmente indican que el disco está lleno o que el medio esta mal.
11H	**@BUSCAR LA PRIMERA COMBINACION (FCB)**
Entrada	AH = 11H DS:DX = dirección del bloque para control de archivo que se debe buscar
Salida	AL = 00H cuando el archivo fue encontrado AL = 0FFH cuando el archivo no fue encontrado

FIGURA A–2 Contenido del bloque para control de archivos (FCB).

Desplazamiento	Contenido
00H	Lector de disco
01H	Nombre de archivo de 8-caracteres
09H	Extensión para archivo de 3-caracteres
0CH	Número de bloque actual
0EH	Tamaño del registro
10H	Tamaño del archivo
14H	Fecha de creación
16H	Espacio reservado
20H	Número de registro actual
21H	Número de registro relativo

Notas	Los caracteres de comodines (? o *) se pueden utilizar para buscar el nombre de un archivo. El caracter de comodín ? iguala a cualquier carácter y el * iguala a cualquier nombre o extensión.
12H	**@BUSCAR LA PROXIMA COMBINACION (FCB)**
Entrada	AH = 12H DS:DX = dirección del bloque de control para archivo que tiene que ser buscado
Salida	AL = 00H cuando el archivo fue encontrado AL = 0FFH cuando el archivo no fue encontrado
Notas	Esta función es usada después de que la función 11H encuentra el primer nombre de archivo que combine.
13H	**@BORRAR EL ARCHIVO USANDO FCB**
Entrada	AH = 13H DS:DX = dirección del bloque para control de archivo que se tiene que borrar
Salida	AL = 00H cuando el archivo fue borrado AL = OFFH cuando ocurrió un error
Notas	Los errores que ocurren más frecuentemente son errores del medio defectuoso.
14H	**@LEER EN FORMA SECUENCIAL (FCB)**
Entrada	AH = 14H DS:DX = dirección del bloque para control de archivo que tiene que ser leído
Salida	AL = 00H cuando es leído exitosamente AL = 01H cuando se alcanza el final del archivo AL = 02H cuando DTA tiene un ajuste de segmento AL = 03H cuando menos de 128 bytes fueron leídos
15H	**@ESCRIBIR EN FORMA SECUENCIAL (FCB)**
Entrada	AH = 15H DS:DX = dirección del bloque para control de archivo que tiene que escribirse
Salida	AL = 00H cuando se escribió con éxito AL = 01H cuando el disco está lleno AL = 02H cuando DTA tuvo un ajuste del segmento
16H	**@CREAR UN ARCHIVO (FCB)**
Entrada	AH = 16H DS:DX = dirección de un bloque para control de un archivo no abierto
Salida	AL = 00H cuando el archivo fue creado AL = 01H cuando el disco está lleno

17H	**@RENOMBRAR UN ARCHIVO (FCB)**
Entrada	AH = 17H DS:DX = dirección de un bloque para control de un archivo modificado
Salida	AL = 00H cuando el archivo es renombrado AL = 01H cuando ocurrió un error
Notas	Refiérase a la Figura A-3 para el FCB modificado utilizado para renombrar un archivo.
18H	**NO ASIGNADO**
19H	**REGRESAR EL LECTOR DE DISCO ACTUAL**
Entrada	AH = 19H
Salida	AL = lector de disco actual
Notas	AL = 00H para el lector de disco A, 01H para el lector de disco B, etc.
1AH	**ESTABLECER EL AREA DE TRANSFERENCIA DEL DISCO**
Entrada	AH = 1AH DS:DX = dirección para el DTA nuevo
Notas	El área de transferencia del disco está normalmente ubicada dentro del prefijo del segmento del programa en la dirección desplazada 80H. El DTA se utiliza por DOS para todas las transferencias de información del disco utilizando bloques de control del archivo.
1BH	**CONSEGUIR LA TABLA DE ASIGNACIONES (FAT) PARA LA UNIDAD PREDETERMINADA**
Entrada	AH = 1BH

FIGURA A–3 Contenido del bloque para control de archivo modificado (FCB).

Desplazamiento	Contenido
00H	Lector de disco
01H	Nombre de archivo de 8-caracteres
09H	Extensión de 3-caracteres
0CH	Número de bloque actual
0EH	Tamaño del registro
10H	Tamaño del archivo
14H	Fecha de creación
16H	Segundo nombre de archivo

Salida	AL = número de sectores por unidades DS:BX = dirección del descriptor del medio CX = tamaño de un sector en bytes DX = número de unidades en un lector de disco
Notas	Refiérase a la Figura A-4 para el formato del byte del descriptor del medio. El registro DS será cambiado por esta función así que asegúrese de grabarlo antes de usar esta función.
1CH	**OBTENER UNA TABLA DE ASIGNACION (FAT) PARA CUALQUIER UNIDAD**
Entrada	AH = 1CH DL = número de lector de disco
Salida	AL = número de sectores por unidad DS:BX = dirección del descriptor del medio CX = tamaño de un sector en bytes DX = número de unidades en el lector de disco
1DH	**NO ASIGNADO**
1EH	**NO ASIGNADO**
1FH	**NO ASIGNADO**
20H	**NO ASIGNADO**
21H	**@LECTURA ALEATORIA UTILIZANDO FCB**
Entrada	AH = 21H DS:DX = dirección del FCB abierto

FIGURA A-4 Contenido del byte descriptivo del medio.

7	6	5	4	3	2	1	0
?	?	?	?	?	?	?	?

Bit 0 = 0 cuando no es de dos lados
= 1 cuando es de dos lados

Bit 1 = 0 cuando no hay ocho sectores por pista
= 1 cuando hay ocho sectores por pista

Bit 2 = 0 cuando no es removible
= 1 cuando es removible

Salida	AL = 00H cuando es leído exitosamente AL = 01H cuando se alcanza el final del archivo AL = 02H cuando el segmento se ajusta AL = 03H cuando menos de 128 bytes son leídos
22H	**@ESCRIBIR DE MANERA ALEATORIA USANDO FCB**
Entrada	AH = 22H DS:DX = dirección del FCB abierto
Salida	AL = 00H cuando fue escrito exitosamente AL = 01H cuando el disco está lleno AL = 02H cuando el segmento se ajusta
23H	**@INDICAR EL NUMERO DE REGISTROS (FCB)**
Entrada	AH = 23H DS:DX = dirección de FCB
Salida	AL = 00H número de registros AL = 0FFH cuando el archivo no se encuentra
24H	**@ESTABLECER EL TAMAÑO RELATIVO DEL REGISTRO (FCB)**
Entrada	AH = 24H DS:DX = dirección de FCB
Notas	Establece el campo del registro al valor contenido en el FCB.
25H	**ESTABLECER EL INTERRUPTOR PARA VECTOR**
Entrada	AH = 25H AL = número del interruptor para el vector DS:DX = dirección del nuevo procedimiento para el interruptor
Notas	Antes de cambiar el interruptor para vector, se sugiere que el interruptor para el vector actual sea primero grabado utilizando la función 35H de DOS. Esto permite un vínculo para que el vector original pueda ser posteriormente restaurado.
26H	**CREAR EL PREFIJO DE SEGMENTO DEL NUEVO PROGRAMA**
Entrada	AH = 26H DX = dirección del segmento del PSP nuevo
Notas	La Figura A-5 muestra la estructura del prefijo del segmento del programa.

FIGURA A–5 Contenido del prefijo del segmento del programa (PSP).

Desplazamiento	Contenido
00H	INT 20H
02H	Parte superior de la memoria
04H	Reservado
05H	Código de operación
06H	Número de bytes en el segmento
0AH	Dirección para terminar (desplazamiento)
0CH	Dirección para terminar (segmento)
0EH	Dirección para Ctrl-break (desplazamiento)
10H	Dirección para Ctrl-break (segmento)
12H	Dirección para error crítico (segmento)
14H	Dirección para error crítico (desplazamiento)
16H	Reservado
2CH	Dirección para el ambiente (segmento)
2EH	Reservado
50H	Solicitud al DOS
52H	Reservado
5CH	Bloque 1 para control de archivo
6CH	Bloque 2 para control de archivo
80H	Distancia de la línea de comando
81H	Línea de comando

27H	@LECTURA ALEATORIA DE BLOQUES DE ARCHIVOS (FCB)
Entrada	AH = 27H CX = el número de registros DS:DX = dirección del FCB abierto
Salida	AL = 00H cuando fue leído exitosamente AL = 01H cuando se alcanza el final del archivo AL = 02H cuando el segmento se ajusta AL = 03H cuando menos de 128 bytes fueron leídos CX = el número de registros leídos

28H	**@ESCRIBIR EN FORMA ALEATORIA BLOQUES DE ARCHIVO (FCB)**
Entrada	AH = 28H CX = el número de registros DS:DX = dirección del FCB abierto
Salida	AL = 00H cuando fue escrito exitosamente AL = 01H cuando el disco está lleno AL = 02H cuando el segmento se ajusta CX = el número de registros escritos
29H	**@LINEA DE COMANDO PARSE (FCB)**
Entrada	AH = 29H AL = esconder parse DS:SI = dirección de FCB DS:DI = dirección de línea de comando
Salida	AL = 00H cuando no son encontrados ningunos caracteres del nombre del archivo AL = 01H cuando son encontrados los caracteres del nombre del archivo AL = 0FFH cuando el especificador del lector del disco fue incorrecto DS:SI = dirección del carácter después del nombre DS:DI = dirección del primer byte de FCB
2AH	**LEER LA FECHA DEL SISTEMA**
Entrada	AH = 2AH
Salida	AL = día de la semana CX = año (1980-2099) DH = mes DL = día del mes
Notas	El día de la semana es codificado como Domingo = 00H a Sábado = 06H. El año es un número binario igual a 1980 hasta 2099.
2BH	**ESTABLECER LA FECHA DEL SISTEMA**
Entrada	AH = 2BH CX = el año (1980-2099) DH = mes DL = día del mes
2CH	**LEER EL TIEMPO DEL SISTEMA**
Entrada	AH = 2CH
Salida	CH = horas (0 – 23) CL = minutos DH = segundos DL = cientos de segundos

2DH	**ESTABLECER EL TIEMPO DEL SISTEMA**
Entrada	AH = 2DH CH = horas CL = minutos DH = segundos DL = cientos de segundos
2EH	**DISCO VERIFICAR ESCRITO**
Entrada	AH = 2EH AL = 00H para desactivar verificar en escrito AL = 01H para activar verificar en escrito
2FH	**LEER AREA DE TRANSFERENCIA DEL DISCO**
Entrada	AH = 2FH
Salida	ES:BX = contiene la dirección de DTA
30H	**LEER EL NUMERO DE LA VERSION DE DOS**
Entrada	AH = 30H
Salida	AH = número fraccionado de la versión AL = número de versión del número entero
Notas	Por ejemplo, el número 3.2 de la versión DOS se compone como 3 en AL y un 14H en AH.
31H	**TERMINAR Y PERMANECER COMO RESIDENTE (TSR)**
Entrada	AH = 31H AL = el código de retorno de DOS DX = número de párrafos para reservar
Notas	Un párrafo es de 16 bytes y el código de salida de DOS será leído al nivel del archivo de lote con ERRORCODE.
32H	**NO ASIGNADO**
33H	**VERIFICAR CTRL-BREAK**



Entrada	AH = 33H AL = 00H para solicitar el ctrl-break actual AL = 01H para cambiar ctrl-break DL = 00H para desactivar ctrl-break DL = 01H para activar ctrl-break
Salida	DL = Estado actual de ctrl-break
34H	**OBTENER DIRECCION DE LA BANDERA InDOS**
Entrada	AH = 34H
Salida	ES:BX dirección de la bandera InDOS
Notas	La bandera InDOS está disponible en las versiones 3.2 o más actuales de DOS e indica la actividad de DOS. Cuando InDOS = 00H, DOS está inactivo o 0FFH cuando DOS está activo.
35H	**LEER EL INTERRUPTOR PARA VECTOR**
Entrada	AH = 35H AL = número del interruptor para vector
Salida	ES:BX = dirección archivada en el vector
Notas	Esta función de DOS es usada con la función 25H para instalar/retirar los identificadores de interruptores.
36H	**DETERMINAR EL ESPACIO LIBRE EN EL DISCO**
Entrada	AH = 36H DL = número del lector de disco
Salida	AX = FFFFH cuando el lector de disco es inválido AX = número de sectores por unidad BX = número de unidades disponibles CX = bytes por sector DX = número de unidades en un lector de disco
Notas	El lector de disco predeterminado es DL = 00H, lector de disco A = 01H, lector de disco B = 02H, etc.
37H	**NO ASIGNADO**
38H	**REGRESAR AL CODIGO DEL PAIS**
Entrada	AH = 38H AL = 00H para el código actual del país BX = código del país de 16-bits DS:DX = dirección del búfer de información

Salida	AX = código de error cuando el restante fue habilitado BX = código para contador DS:DX = dirección del búfer de información
39H	**CREAR SUBDIRECTORIO**
Entrada	AH = 39H DS:DX = dirección para el nombre del subdirectorio de la cadena ASCII
Salida	AX = código de error cuando el restante está activado
Notas	Cadena ASCII es el nombre del subdirectorio en el código ASCII terminado con 00H en vez de un regreso de carro/alimentación de línea.
3AH	**BORRAR SUBDIRECTORIO**
Entrada	AH = 3AH DS:DX = dirección del nombre del subdirectorio de la cadena ASCII
Salida	AX = código de error cuando el restante fue activado
3BH	**CAMBIAR SUBDIRECTORIO**
Entrada	AH = 3BH DS:DX = dirección del nombre nuevo del subdirectorio de la cadena ASCII-Z
Salida	AX = código de error cuando el restante está activado
3CH	**CREAR UN ARCHIVO NUEVO**
Entrada	AH = 3CH CX = palabra de atributo DS:DX = dirección del nombre de archivo de la cadena ASCII-Z
Salida	AX = código de error cuando el restante está activado AX = identificador de archivo cuando el restante fue eliminado
Notas	La palabra de atributo puede contener cualquiera de ios siguientes (sumados): 01H acceso sólo de lectura, 02H = archivo o directorio escondido, 04H archivo del sistema, 08H = nivel del volumen, 10H = subdirectorio, y 20H = bit de archivo. En la mayoría de los casos un archivo es creado con 0000H.
3DH	**ABRIR UN ARCHIVO**
Entrada	AH = 3DH AL = código de acceso DS:DX = dirección del nombre de archivo de la cadena ASCII-Z
Salida	AX = código de error cuando el restante está activado AX = identificador de archivo cuando el restante fue eliminado

Notas	El código de acceso en AL = 00H para acceso de sólo lectura, AL = 01H para acceso de sólo escribir, y AL = 02H para acceso de leer/escribir. Para los archivos compartidos en un ambiente de red, bit 4 de AL = 1 negará el acceso de leer/escribir, bit 5 de AL = 1 negará un acceso a escribir, bits 4 y 5 de AL = 1 negarán el acceso a leer, bit 6 de AL = 1 no niega ninguno, bit 7 de AL = 0 ocasiona que el archivo sea heredado por el hijo y bit 7 de AL = 1 está limitado al proceso actual.
3EH	**CERRAR UN ARCHIVO**
Entrada	AH = 3EH BX = identificador de archivo
Salida	AX = código de error cuando el restante está activado
3FH	**LEER UN ARCHIVO**
Entrada	AH = 3FH BX = identificador de archivos CX = número de bytes que deben leerse DS:DX = dirección del búfer del archivo para mantener los datos al leer
Salida	AX = código de error cuando el restante está activado AX = número de bytes leídos cuando el restante está eliminado
40H	**ESCRIBIR UN ARCHIVO**
Entrada	AH = 40H BX = identificador de archivos CX = número de bytes para escribirse DS:DX = dirección del búfer para archivo que sostiene los datos escritos
Salida	AX = código de error cuando el restante está activado AX = número de bytes escritos cuando el restante está eliminado
41H	**ELIMINAR UN ARCHIVO**
Entrada	AH = 41H DS:DX = dirección del nombre de archivo de la cadena ASCII
Salida	AX = código de error cuando el restante está activado
42H	**MOVER EL APUNTADOR DEL ARCHIVO**
Entrada	AH = 42H AL = técnica de movimiento BX = identificador de archivos CX:DX = número de bytes que el apuntador movió

Salida	AX = código de error cuando el restante está activado AX:DX = apuntador para bytes movido
Notas	La técnica de movimiento ocasiona que el apuntador se mueva del principio del archivo cuando AL = 00H, de la localidad actual si AL = 01H y del final del archivo si AL = 02H. La cuenta será guardada para que DX contenga los 16-bits menos significantes y CX o AX contienen los 16 bits más importantes.
43H	**ATRIBUTOS DE LEER/ESCRIBIR DEL ARCHIVO**
Entrada	AH = 43H AL = 00H para leer atributos AL = 01H para escribir atributos CX = palabra de atributo (véase la función 3CH) DS:DX = dirección del nombre del archivo de la cadena ASCII-Z
Salida	AX = código de error cuando el restante está activado CX = palabra de atributo del restante eliminado
44H	**CONTROL DEL PERIFERICO I/O (IOTCL)**
Entrada	AH = 44H AL = código (véase notas) AL = 01H para escribir atributos BX = identificador de archivos o número de periférico CX = número de bytes DS:DX = datos o dirección
Salida	X = código de error cuando el restante está activado AX y DX = parámetros
Notas	Los códigos encontrados en AL son los siguientes: 00H = leer el estado del periférico (DX = estado) 01H = escribir el estado del periférico (DX = estado escrito) 02H = leer información del periférico (DS:DX = dirección del búfer) 03H = escribir información al periférico (DS:DX = dirección del búfer) 04H = leer información del lector del disco 05H = escribir información al lector del disco 06H = leer el estado de la entrada de información (AL = 00H listo o 0FH no listo) 07H = leer el estado de la salida de información (AL = 00H listo o 0FH no listo) 08H = ¿medio removible? (AL = 00H removible, 01H fijo) 09H = ¿periférico local o remoto? (bit 12 del juego DX para remoto) 0AH = ¿identificador local o remoto? (bit 15 del juego DX para remoto) 0BH = cambiar la cuenta de la entrada 0CH = control del I/O genérico para los periféricos de caracteres 0DH = control del I/O genérico para los periféricos de bloque 0EH = número de salida de los periféricos lógicos (AL = número) 0FH = cambiar el número de los periféricos lógicos
45H	**DUPLICAR EL IDENTIFICADOR DE ARCHIVOS**
Entrada	AH = 45H BX = identificador de archivos actual

Salida	AX = código de error cuando el restante está activado AX = duplicar identificador de archivos
46H	**FORZAR IDENTIFICADORES DUPLICADOS DE ARCHIVOS**
Entrada	AH = 46H BX = identificador de archivos actual CX = identificador de archivos nuevo
Salida	AX = código de error cuando el restante está activado
Notas	Esta función trabaja como la función 45H excepto que la función 45H le permite a DOS seleccionar al identificador nuevo mientras que esta función le permite al usuario seleccionar el identificador nuevo.
47H	**LEER DIRECTORIO ACTUAL**
Entrada	AH = 47H DL = número de lector de disco DS:SI = dirección de un búfer de 64 bytes para el nombre del directorio
Salida	DS:SI direcciona el nombre del directorio actual cuando el restante fue eliminado
48H	**ASIGNAR EL BLOQUE DE MEMORIA**
Entrada	AH = 48H BX = número de párrafos para asignar CX = identificador de archivos nuevo
Salida	BX = el bloque más grande disponible si el restante fue eliminado
49H	**LIBERAR EL BLOQUE DE MEMORIA ASIGNADO**
Entrada	AH = 49H ES = segmento de la dirección del bloque que tiene que liberarse CX = identificador de archivos nuevo
Salida	El restante indica un error cuando sea establecido
4AH	**MODIFICAR EL BLOQUE DE MEMORIA ASIGNADO**
Entrar	AH = 4AH BX = nuevo tamaño del bloque en párrafos ES = dirección del segmento del bloque para ser modificado
Salida	BX = bloque más grande disponible cuando el restante fue eliminado.

4BH	**CARGAR O EJECUTAR UN PROGRAMA**
Entrada	AH = 4BH AL = código de función ES:BX = dirección del bloque para parámetro DS:DX = dirección del comando de la cadena ASCII-Z
Salida	El restante indica un error cuando fue activado
Notas	Los códigos de las funciones son AL = 00H para cargar y ejecutar un programa y AL = 03H para cargar un programa pero no ejecutarlo. La Figura A-6 muestra el bloque de parámetros usado con esta función.
4CH	**ELIMINAR UN PROCESO**
Entrada	AH = 4CH AL = código de error
Salida	Regresa el control a DOS
Notas	Esta función regresa el control a DOS con el código de error grabado para que se pueda obtener utilizando el sistema de procesamiento de lote de DOS ERROR LEVEL. Normalmente usamos esta función con un código de error de 00H para regresar a DOS.
4DH	**LEER EL CODIGO DE SALIDA**
Entrada	AH = 4DH

FIGURA A–6 Los bloques para los parámetros utilizados con la función 4BH (EXEC). (a) Para el código de función 00H. (b) Para el código de función 03H.

(a)

Desplazamiento	Contenido
00H	Dirección para el ambiente (segmento)
02H	Dirección para la línea de comando (desplazamiento)
04H	Dirección para la línea de comando (segmento)
06H	Dirección para el bloque 1 para el control del archivo (desplazamiento)
08H	Dirección para el bloque 1 para control del archivo (segmento)
0AH	Dirección para el bloque 2 para control del archivo (segmento)
0CH	Dirección para el bloque 2 para control del archivo (desplazamiento)

(b)

Desplazamiento	Contenido
00H	Dirección para el segmento destino sobrepuesto
02H	Factor de reubicación

Salida	AX = indica el código de error
Notas	Esta función es usada para obtener un código de estado de salida creado ejecutando un programa con la función 4BH de DOS. Los códigos de salida son: AX = 0000H para una terminación normal—sin errores. AX = 0001H para una terminación de ctrl-break, AX = 0002H para un error crítico de periférico y AX = 0003H para una terminación por un INT 31H.
4EH	**ENCONTRAR EL PRIMER ARCHIVO QUE COMBINE**
Entrada	AH = 4EH CX = atributos del archivo DS:DX = dirección del nombre del archivo de la cadena ASCII-Z
Salida	El restante está habilitado para un archivo no encontrado
Notas	Esta función buscará en el directorio actual u otro para el primer archivo que combine. A la salida, el DTA contiene la información del archivo. Vea la Figura A-7 para el área de transferencia del disco (DTA).
4FH	**ENCONTRAR EL PROXIMO ARCHIVO QUE COMBINE**
Entrada	AH = 4FH
Salida	El restante está activado para un archivo no encontrado
Notas	Esta función es usada después de que el primer archivo fue encontrado con la función 4EH.
50H	**ESTABLECER LA DIRECCION DEL PREFIJO DEL SEGMENTO DEL PROGRAMA (PSP)**
Entrada	AH = 50H BX = dirección desplazada del PSP nuevo
Notas	Un cuidado excesivo se debe usar con esta función porque no es posible ninguna recuperación de error.

FIGURA A-7 Area para transferencia de información (DTA) usada para encontrar un archivo.

Desplazamiento	Contenido
15H	Atributos
16H	Tiempo para creación
18H	Fecha de creación
1AH	Tamaño del archivo de palabra inferior
1CH	Tamaño del archivo de palabra superior
1EH	Buscar nombre del archivo

51H	**OBTENER LA DIRECCION DE PSP**
Entrada	AH = 51H
Salida	BX = dirección actual del segmento PSP
52H	**NO ASIGNADO**
53H	**NO ASIGNADO**
54H	**LEER EL ESTADO DE VERIFICACION DEL DISCO**
Entrada	AH = 54H
Salida	AL = 00H cuando verificar no está activada AL = 01H cuando verificar está activada
55H	**NO ASIGNADO**
56H	**RENOMBRAR ARCHIVO**
Entrada	AH = 56H ES:DI = dirección de la cadena ASCII-Z conteniendo el nuevo nombre del archivo DS:DX = dirección de la cadena ASCII-Z que contiene el archivo que se tiene que renombrar
Salida	El restante está activado para la condición de error
57H	**LEER LA MARCA DE FECHA Y TIEMPO DEL ARCHIVO**
Entrada	AH = 57H AL = código de función BX = identificador de archivos CX = nuevo tiempo DX = nueva fecha
Salida	El restante está activado para la condición de error CX = hora cuando el restante fue eliminado DX = fecha cuando el restante fue eliminado
Notas	AL = 00H para leer la fecha y el tiempo o 01H para escribir la fecha y tiempo.
58H	**NO ASIGNADO**

59H	OBTENER LA INFORMACION DE ERROR EXTENDIDA
Entrada	AH = 59H BX = 0000H para la versión 3.X de DOS
Salida	AX = código de error extendido BH = clase de error BL = acción recomendada CH = locus
Notas	A continuación están los códigos de error encontrados en AX: 0001H = número de función inválido 0002H = archivo no encontrado 0003H = ruta de acceso no encontrada 0004H = ningunos identificadores de archivos están disponibles 0005H = acceso negado 0006H = identificador de archivos inválido 0007H = falla del bloque de control de la memoria 0008H = memoria insuficiente 0009H = dirección inválida del bloque de memoria 000AH = falla en el ambiente 000BH = formato inválido 000CH = código de acceso inválido 000DH = información inválida 000EH = unidad desconocida 000FH = lector de disco inválido 0010H = intentado para quitar el directorio actual 0011H = no el mismo periférico 0012H = no más archivos 0013H = disco con protección contra escribir 0014H = unidad desconocida 0015H = lector de disco no listo 0016H = comando desconocido 0017H = error de información (CRC revisar error) 0018H = largo de la estructura de una petición mala 0019H = buscar error 001AH = tipo de medio desconocido 001BH = sector no encontrado 001CH = impresora fuera de papel 001DH = escribir falla 001EH = leer falla 001FH = falla general 0020H = compartir violación 0021H = cerrar violación 0022H = cambio de disco inválido 0023H = FCB no disponible 0024H = exceso del búfer compartido 0025H = incompatibilidad de la página del código 0026H = manejar la operación del final del archivo no terminada 0027H = disco lleno 0028H —0031H reservado 0032H = requerimiento de red no apoyado 0033H = máquina remota no mencionada 0034H = nombre duplicado en la red 0035H = nombre de la red no encontrado 0036H = red ocupada 0037H = periférico ya no existe en la red 0038H = límite del comando netBIOS excedido

0039H = error en el hardware para el adaptador de la red
003AH = respuesta incorrecta de la red
003BH = error inesperado en la red
003CH = adaptador remoto es incompatible
003DH = cola de espera para imprimir está llena
003EH = no hay suficiente espacio para imprimir el archivo
003FH = archivo para impresión fue borrado
0040H = nombre de la red borrado
0041H = acceso a la red negado
0042H = tipo incorrecto del periférico de la red
0043H = nombre de la red no encontrado
0044H = nombre de la red excedió el límite
0045H = límite de la sesión netBIOS excedido
0046H = pausa temporal
0047H = requerimiento de la red no aceptado
0048H = pausa de impresión o de redireccionamiento del disco
0049H — 004FH reservado
0050H = archivo ya existe
0051H = duplicar FCB
0052H = no se puede hacer el directorio
0053H = falla en INT 24H (error crítico)
0054H = demasiados redireccionamientos
0055H = duplicar redireccionamiento
0056H = clave inválida
0057H = parámetro inválido
0058H = falla al escribir en la red
0059H = función no apoyada por la red
005AH = componente del sistema requerido no instalado
0065H = periférico no instalado

A continuación están los códigos de clase error como se encuentran en BH:

01H = no hay recursos disponibles
02H = error transitorio
03H = error de autorización
04H = error del software interno
05H = error del hardware
06H = falla en el sistema
07H = error de aplicación del software
08H = elemento no encontrado
09H = formato inválido
0AH = elemento bloqueado
0BH = error del medio
0CH = elemento ya existe
0DH = error no conocido

A continuación está la acción recomendada como se encuentra en BL:

01H = reintente la operación
02H = demorar y reintente la operación
03H = reintento del usuario
04H = abortar el procesamiento
05H = salida inmediata
06H = ignorar error
07H = reintentar con la intervención del usuario

A continuación está una lista de locus en CH:

01H = fuente desconocida
02H = bloquear el error del periférico

	03H = área de la red 04H = error del periférico serial 05H = error de la memoria
5AH	**CREAR UN NOMBRE UNICO PARA ARCHIVO**
Entrada	AH = 5AH CX = código de atributo DS:DX = dirección de la ruta de acceso del directorio de la cadena ASCII-Z
Salida	El restante está activado para una condición de error AX = identificador de archivo cuando el restante fue eliminado DS:DX = dirección del nombre de directorio agregada
Notas	La ruta de acceso del directorio del archivo ASCII-Z debe terminar con una diagonal invertida (\). A la salida, el nombre del directorio será agregado como un nombre único de archivo.
5BH	**CREAR UN ARCHIVO DOS**
Entrada	AH = 5BH CX = código de atributo DS:DX = dirección de la cadena ASCII-Z contiene el nombre del archivo
Salida	El restante está activado para la condición de error AX = identificador de archivo cuando el restante fue eliminado
Notas	La función sólo trabaja en la versión 3.X o más de DOS.
5CH	**CERRAR/ABRIR EL CONTENIDO DEL ARCHIVO**
Entrada	AH = 5CH BX = identificador de archivos CX:DX = dirección desplazada del área cerrada/abierta SI:DI = número de bytes para cerrar o abrir empezando en el desplazamiento
Salida	El restante está activado para la condición de error.
5DH	**ESTABLECER INFORMACION DE ERROR EXTENDIDO**
Entrada	AH = 5DH AL = 0AH DS:DX = dirección de la estructura de información del error extendido
Notas	Esta función es usada por la versión 3.1 o más de DOS para archivar la información de error extendido.
5EH	**RED/IMPRESORA**
Entrada	AH = 5EH AL = 00H (obtener el nombre de la red) DS:DX = dirección de la cadena ASCII-Z que contiene el nombre de la red

Salida	El restante está activado para la condición de error CL = número netBIOS cuando el restante está eliminado
Entrada	AH = 5EH AL = 02H (definir la impresora de la red) BX = lista de redireccionamiento CX = distancia de la cadena para la instalación DS:DX = dirección del búfer para establecer la impresora
Salida	El restante está activado para una condición de error
Entrada	AH = 5EH AL = 03H (leer la cadena de establecimiento de la impresora de la red) BX = lista de redireccionamiento DS:DX = dirección del búfer para establecer la impresora
Salida	El restante está activado para la condición de error CX = distancia de la cadena de instalación cuando el restante fue eliminado ES:DI = dirección del búfer para establecer la impresora
62H	**OBTENER LA DIRECCION PSP**
Entrada	AH = 62H
Salida	BX = dirección del segmento del programa actual
Notas	La función sólo trabaja en la versión 3.0 o más de DOS
65H	**OBTENER INFORMACION DEL PAIS EXTENDIDA**
Entrada	AH = 65H AL = código de función ES:DI = dirección del búfer que va a recibir la información
Salida	El restante está activado para condición de error CX = distancia de la información del país
Notas	La función sólo trabaja con la versión 3.3 o más de DOS.
66H	**OBTENER/ESTABLECER PAGINA DE CODIGO**
Entrada	AH = 66H AL = código de función BX = número de la página del código
Salida	El restante está activado para una condición de error BX = número de la página de código activo DX = número de la página de código predeterminado
Notas	Un código de función en AL de 01H obtiene el número de la página del código y un código de 02H establece el número de la página del código.

67H	**ESTABLECER LA CUENTA DEL IDENTIFICADOR**
Entrada	AH = 67H BX = número de identificadores deseados
Salida	El restante está activado para la condición de error
Notas	Esta función está disponible para la versión 3.3 o más de DOS.

68H	**ARCHIVO COMPROMETIDO**
Entrada	AH = 68H BX = número de identificador
Salida	El restante está activado para la condición de error. De otra manera, la marca de la fecha y la hora será escrita al directorio
Notas	Esta función está disponible para la versión 3.3 o más de DOS.

6CH	**ARCHIVO ABIERTO EXTENDIDO**
Entrada	AH = 6CH AL = 00H BX = modo abierto CX = atributos DX = abrir bandera DS:SI = dirección del nombre del archivo de la cadena ASCII-Z
Salida	AX = código de error cuando el restante está establecido AX = identificador cuando el restante fue eliminado CX = archivo 0001H existía y fue abierto CX = archivo 0002H no existía y fue creado
Notas	Esta función está disponible en la versión 4.0 o más de DOS.

SOLICITUDES A FUNCIONES DE BIOS

Además de la solicitud de función INT 21H de DOS, algunas otras solicitudes de funciones de BIOS son útiles para controlar al ambiente I/O de la computadora. A diferencia de INT 21H, el cual existe en el programa DOS, las solicitudes de funciones de BIOS se encuentran archivadas en el BIOS ROM. Estas funciones BIOS controlan directamente a los periféricos de I/O con o sin DOS cargado en un sistema.

INT 10H

El interruptor INT 10H BIOS frecuentemente se llama el interruptor para los servicios de video porque controla directamente la pantalla de video en un sistema. La instrucción INT 10H utiliza un registro AH para seleccionar el servicio de video proporcionado por este interruptor.

Selección del modo de video. El modo de operación para la pantalla de video es seleccionado colocando un 00H en AH seguido por uno de los muchos números de modo en AL. La Tabla A-3 menciona los modos de operación encontrados en los sistemas de pantalla de video utilizando modos de video estándares. El VGA puede usar cualquier modo mencionado, mientras que las otras pantallas son más estrictas en su uso. Modos adicionales de resolución más altos son explicados más adelante en esta sección.

El Ejemplo A-6 muestra una secuencia corta de instrucciones que colocan a la pantalla de video en el modo 03H. Este modo está disponible en pantallas CGA, EGA y VGA. También permite que la pantalla dibuje una prueba con 16 colores en varias resoluciones dependiendo del adaptador para la pantalla.

EJEMPLO A-6

```
0000 B4 00        MOV   AH,0        ;seleccione el modo
0002 B0 03        MOV   AL,3        ;el modo es 03H
0004 CD 10        INT   10H
```

TABLA A-3 Modos de las pantallas para video

Modo	*Tipo*	*Columnas*	*Hileras*	*Resolución*	*Estándar*	*Colores*
00H	Texto	40	25	320 × 200	CGA	2
00H	Texto	40	25	320 × 350	EGA	2
00H	Texto	40	25	360 × 400	VGA	2
01H	Texto	40	25	320 × 200	CGA	16
01H	Texto	40	25	320 × 350	EGA	16
01H	Texto	40	25	360 × 400	VGA	16
02H	Texto	80	25	640 × 200	CGA	2
02H	Texto	80	25	640 × 350	EGA	2
02H	Texto	80	25	720 × 400	VGA	2
03H	Texto	80	25	640 × 200	CGA	16
03H	Texto	80	25	640 × 350	EGA	16
03H	Texto	80	25	720 × 400	VGA	16
04H	Gráficas	40	25	320 × 200	CGA	4
05H	Gráficas	40	25	320 × 200	CGA	2
06H	Gráficas	80	25	640 × 200	CGA	2
07H	Texto	80	25	720 × 350	EGA	4
07H	Texto	80	25	720 × 400	VGA	4
0DH	Gráficas	80	25	320 × 200	CGA	16
0EH	Gráficas	80	25	640 × 200	CGA	16
0FH	Gráficas	80	25	640 × 350	EGA	4
10H	Gráficas	80	25	640 × 350	EGA	16
11H	Gráficas	80	30	640 × 480	VGA	2
12H	Gráficas	80	30	640 × 480	VGA	16
13H	Gráficas	40	25	320 × 200	VGA	256

Control del cursor. La Tabla A-4 muestra los códigos de función utilizados para controlar el cursor en la pantalla de video. Esta función de control del cursor funcionará en cualquier pantalla de video desde la pantalla CGA hasta la pantalla VGA más actual.

Si un adaptador SVGA (Super VGA) EVGA (VGA extendido), o XVGA (también VGA extendido) está disponible, el modo VGA super es establecido utilizando la solicitud de función AX = 4F02H de INT 10H con BX = al modo VGA para estos adaptadores de pantalla avanzados. Esto conforma al estándar VESA para los adaptadores VGA. La Tabla A-5 muestra los modos seleccionados por el registro BX para esta solicitud de función de INT 10H.

TABLA A-4 Funciones de video BIOS (INT 10H)

00H	**SELECCIONAR EL MODO DE VIDEO**
Entrada	AH = 00H AL = número de modo
Salida	Modo cambiado y la pantalla borrada
01H	**SELECCIONAR TIPO DE CURSOR**
Entrada	AH = 01H CH = número de la línea para iniciar CL = número de la última línea
Salida	Tamaño del cursor modificado
02H	**SELECCIONAR POSICION DEL CURSOR**
Entrada	AH = 02H BH = número de página (normalmente 0) DH = número de hilera (comenzando con 0) DL = número de columna (comenzando con 0)
Salida	Cambia el cursor a una nueva posición
03H	**LEER POSICION DEL CURSOR**
Entrada	AH = 03H BH = número de página
Salida	CH = línea inicial (tamaño del cursor) CL = línea final (tamaño del cursor) DH = hilera actual DL = columna actual
04H	**LEER LAPIZ DE LUZ**
Entrada	AH = 04H (no apoyado en VGA)

TABLA A–4 *(Continuación)*

Salida	AH = 0, activa el lápiz de luz BX = columna de pixels CX = hilera de pixels DH = hilera de caracteres DL = columna de caracteres
05H	**SELECCIONAR PAGINA DE LA PANTALLA**
Entrada	AH = 05H AL = número de página
Salida	Número de página seleccionado. A continuación están los números de página válidos. Modo 0 y 1 apoyan las páginas 0-7 Modo 2 y 3 apoyan las páginas 0-7 Modo 4, 5 y 6 apoyan la página 0 Modo 7 y D apoyan las páginas 0-7 Modo E apoya las páginas 0-3 Modo F y 10 apoyan las páginas 0-1 Modo 11, 12 y 13 apoyan la página 0
06H	**BUSCAR CON AVANCE PAGINA**
Entrada	AH = 06H AL = número de líneas para buscar (0 borra la ventana) BH = atributo de caracteres para las nuevas líneas CH = hilera superior para buscar en la ventana CL = columna izquierda de la ventana para buscar DH = última hilera de la ventana para buscar DL = columna derecha de la ventana para buscar
Salida	Desplaza la ventana de abajo hasta arriba de la pantalla. Líneas en blanco llenan la parte inferior utilizando el atributo de caracter en BH.
07H	**BUSCAR CON RETROCESO PAGINA**
Entrada	AH = 07H AL = número de líneas para buscar (0 borra a ventana) BH = atributo del carácter para las líneas nuevas CH = hilera superior para de la ventana para buscar CL = columna izquierda para la ventana para buscar DH = última hilera de la ventana para buscar DL = columna derecha de la ventana para buscar
Salida	Desplaza la ventana de arriba a abajo de la pantalla. Líneas en blanco llenan desde arriba utilizando el atributo del carácter en BH.
08H	**LEER ATRIBUTO/CARACTER EN LA POSICION ACTUAL DEL CURSOR**
Entrada	AH = 08H BH = número de la página

TABLA A–4 *(Continuación)*

Salida	AL = código del carácter ASCII AH = atributo del carácter Nota: Esta función no avanza al cursor.
09H	**ESCRIBIR ATRIBUTO/CARACTER EN LA POSICION ACTUAL DEL CURSOR**
Entrada	AH = 09H AL = código del carácter ASCII BH = número de la página BL = atributo del carácter CX = número de caracteres para escribir
Salida	Nota: Esta función no avanza el cursor.
0AH	**ESCRIBIR CARACTER EN LA POSICION ACTUAL DEL CURSOR**
Entrada	AH = 0AH AL = código del caracter ASCII BH = número de la página CX = número de caracteres para escribir
Salida	Nota: Esta función no avanza el cursor.
0FH	**LEER EL MODO DE VIDEO**
Entrada	AH = 0FH
Salida	AL = modo de video actual AH = número de columnas de caracteres BH = número de la página
10H	**ESTABLECER EL REGISTRO DE LA PALETA VGA**
Entrada	AH = 10H AL = 10H BX = número de color (0-255) CH = verde (0-63) CL = azul (0-63) DH = rojo (0-63)
Salida	El color del registro en paleta será cambiado. Nota: los primeros 16 colores (0-15) son utilizados en el modo de texto VGA de 16 colores y otros modos.
10H	**LEER EL REGISTROS DE LA PALETA VGA**
Entrada	AH = 10H AL = 15H BX = número de color (0-255)

TABLA A–4 *(Continuación)*

Salida	CH = verde CL = azul DH = rojo
11H	**OBTENER EL CONJUNTO DE CARACTERES ROM**
Entrada	AH = 11H AL = 30H BH = 2 = conjunto de caracteres 8 X 14 de ROM BH = 3 = conjunto de caracteres 8 X 8 de ROM BH = 4 = conjunto de caracteres extendido 8 X 8 de ROM BH = 5 = conjunto de caracteres 9 X 14 de ROM BH = 6 = conjunto de caracteres 8 X 16 de ROM BH = 7 = conjunto de caracteres 9 X 16 de ROM
Salida	CX = bytes por carácter DL = hileras por carácter ES:BP = dirección del conjunto de caracteres

TABLA A–5 Funciones extendidas de VGA

BX	*Función*
100H	640 X 400 con 256 colores
101H	640 X 480 con 256 colores
102H	800 X 600 con 16 colores
103H	800 X 600 con 256 colores
104H	1,024 X 768 con 16 colores
105H	1,024 X 768 con 256 colores
106H	1,280 X 1,024 con 16 colores
107H	1,280 X 1,024 con 256 colores
108H	80 X 60 en el modo texto
109H	132 X 25 en el modo texto
10AH	132 X 43 en el modo texto
10BH	132 X 50 en el modo texto
10CH	132 X 60 en el modo texto

INT 11H

Esta función se usa para determinar el tipo de equipo instalado en el sistema. Para utilizar esta solicitud, el registro AX se carga con FFFFH y después la instrucción INT 11H se ejecuta. A cambio, INT 11H proporciona información como se menciona en la figura A-8.

INT 12H

El tamaño de la memoria se rehabilita por la instrucción INT 12H. Después de ejecutar la instrucción INT 12H, el registro AX contiene el número de bloques de memoria de 1K byte (memoria convencional en los primeros 1M bytes de espacio de dirección) instalada en la computadora.

FIGURA A-8 El contenido de AX como indica el equipo anexado a la computadora.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P1	P0		G	S2	S1	S0	D2	D1							

P1, P0 = número de puertos paralelos
G = 1 cuando el juego I/O esta conectado
S2, S1, S0 = número de puertos seriales
D2, D1 = número de lectores de disco

INT 13H

Esta solicitud controla los diskettes (5 1/4" o 3 1/2") y también los lectores de disco fijos o duros conectados al sistema. La Tabla A-6 muestra las funciones disponibles para este interruptor vía el registro AH. El control directo de un disco floppy o disco duro puede llevar a problemas. Por lo tanto, sólo proporcionamos una lista de las funciones sin detalles sobre su uso. Antes de utilizar estas funciones, refiérase a la literatura de BIOS disponible de la compañía que produjo su versión de BIOS ROM.

INT 14H

El interruptor 14H controla los puertos seriales de COM (comunicaciones) conectados a la computadora. El sistema de la computadora contiene dos puertos COM, COM1 y COM2, al menos que usted tenga una máquina de tipo AT más nueva en donde el número de puertos para comunicaciones están extendidos a COM3 y COM4. Los puertos para comunicaciones son normalmente controlados con los paquetes de software que permiten la transferencia de información por medio de un modem y las líneas telefónicas. La instrucción INT 14H controla estos puertos como se muestra en la Tabla A-7.

INT 15H

La instrucción INT 15H controla muchos de los varios periféricos de I/O conectados a la computadora. También permite acceso a la operación del modo protegido y al sistema de la memoria extendida en un sistema 80286, 80386 o 80486. La Tabla A-8 muestra las funciones apoyadas por INT 15H.

INT 16H

La instrucción INT 16H se usa como un interruptor para el teclado. A este interruptor se tiene acceso por un interruptor INT 21H de DOS, pero se puede tener acceso directamente. La Tabla A-9 muestra las funciones realizadas por INT 16H.

INT 17H

La instrucción INT 17H tiene acceso al puerto de la impresora paralela normalmente llamado LPT1 en la mayoría de los sistemas. La Tabla A-10 muestra las tres funciones disponibles para la instrucción INT 17H.

TABLA A–6 Función I/O del disco vía INT 13H

AH	Función
00H	Reiniciar el sistema de disco
01H	Colocar el estado del disco en AL
02H	Leer sector
03H	Escribir sector
04H	Verificar sector
05H	Formatear pista
06H	Formatear pista dañada
07H	Formatear unidad de disco
08H	Obtener los parámetros de unidad de disco
09H	Iniciar las características del disco fijo
0AH	Leer el sector largo
0BH	Escribir el sector largo
0CH	Buscar
0DH	Reiniciar el sistema del disco fijo
0EH	Leer el sector para el búfer
0FH	Escribir el sector para el búfer
10H	Obtener el estado de la unidad de disco
11H	Recalibrar la unidad de disco
12H	Diagnósticos del controlador RAM
13H	Diagnósticos del controlador de la unidad de disco
14H	Diagnósticos internos del controlador
15H	Obtener el tipo de disco
16H	Obtener el estado del cambio de disco
17H	Establecer el tipo de disco
18H	Establecer el tipo de medio
19H	Estacionar cabezas
1AH	Formatear la unidad de disco ESDI

TABLA A–7 Interruptor INT 14H para el puerto COM

AH	Función
00H	Iniciar el puerto de comunicaciones
01H	Enviar carácter
02H	Recibir carácter
03H	Obtener el estado del puerto COM
04H	Puerto extendido para inicialización de comunicaciones
05H	Control extendido del puerto de comunicaciones

TABLA A–8 El interruptor INT 15H del subsistema I/O

AH	*Función*
00H	Motor del cassette encendido
01H	Motor del cassette apagado
02H	Leer cassette
03H	Escribir cassette
0FH	Formatear el interruptor periódico de la unidad de disco ESDI
21H	Interruptor del teclado
80H	Periférico abierto
81H	Periférico cerrado
82H	Terminar proceso
83H	Esperar evento
84H	Leer joystick
85H	Tecla de requerimiento del sistema
86H	Retraso
87H	Mover el bloque extendido de la memoria
88H	Obtener el tamaño de la memoria extendida
89H	Entrar el modo protegido
90H	Espera del periférico
91H	Autoprueba al encender del periférico (POST)
C0H	Obtener el ambiente del sistema
C1H	Obtener la dirección del área de información extendida de BIOS
C2H	Apuntador del mouse
C3H	Establecer el medidor de tiempo de vigilancia
C4H	Selección de opción programable

TABLA A–9 Interruptor para teclado INT 16H

AH	*Función*
00H	Leer el carácter del teclado
01H	Obtener el estado del teclado
02H	Obtener las banderas del teclado
03H	Establecer el rango de repetición
04H	Establecer el toque del teclado
05H	Empujar el código de carácter y examinar

TABLA A–10 Interruptor para impresora paralela INT 17H

AH	*Función*
00H	Imprimir carácter
01H	Iniciar impresora
02H	Obtener el estado de la impresora

MAPA DE LA MEMORIA DEL SISTEMA DOS

La figura A-9 muestra el mapa de la memoria usado por un sistema de computadora DOS. El primer 1M byte de memoria es mostrado con todas las áreas que contienen diferentes periféricos y programas utilizados con DOS. El área temporal de programa (TPA) es en donde los programas de las aplicaciones de DOS son cargados y ejecutados. El tamaño del TPA es normalmente un poco más que los 500K bytes al menos que muchos programas y controladores TSR llenen la memoria antes que el TPA.

LAS ASIGNACIONES DE MEMORIA BAJA DE DOS

La Tabla A-11 muestra las asignaciones de memoria baja (00000H-005FFH) para el sistema microprocesador utilizando DOS. Esta área de la memoria contiene al interruptor para vectores, área de información BIOS y el área de información DOS/BIOS mostrada en la figura A-9.

MAPA DE LA MEMORIA DE LA VERSION 5.0 DE DOS

La versión 5.0 de Microsoft DOS tiene un mapa de la memoria un poco diferente que las versiones anteriores de DOS debido a su capacidad de cargar controladores y programas en el área del sistema. Cuando el microprocesador es un 80386 o 80486, la memoria entre la memoria ROM ubicada entre las direcciones A0000H y FFFFFH puede ser rellena con memoria extendida para controladores y programas. En muchas áreas de memoria de sistemas D0000H-DFFFFH no es utilizada como lo es E0000H-EFFFFH. Estas áreas pueden llenarse con memoria extendida por medio de paginación de memoria encontrada en los microprocesadores 80386 y 80486. Esta nueva área de la memoria se puede entonces llenar con una dirección para programas de memoria normales del modo real extendiendo la memoria disponible a las aplicaciones de DOS.

Los controladores HIMEM.SYS y EMM386.SYS son utilizados para realizar el relleno. Cuando desea usar el área de la memoria E0000H-EFFFFH, deberá cargar EMM386.SYS como

FIGURA A-9 Mapa de la memoria de DOS mostrando los primeros 1M bytes de memoria.

TABLA A–11 Asignaciones de la memoria baja de DOS

Localidad	*Propósito*	
00000H-002FFH	Interruptores para vectores del sistema	
00300H-003FFH	Interruptores para vectores del sistema, encendido y área de arranque	
00400H-00407H	Direcciones base del puerto I/O de COM1-COM4	
00408H-0040FH	Direcciones base del puerto I/O de LP1-LPT4	
00410H-00411H	Palabra para bandera de equipo, colocada en AX por un INT 11H	
	Bit	Propósito
	15–14	Número de impresoras paralelas (LPT1-LPT4)
	13	MODEM interno instalado
	12	Joystick instalado
	11–9	Número de puertos seriales (COM1-COM4)
	8	No usado
	7–6	Número de lectores de disco
	5–4	Modo de video
	3–2	No usado
	1	Coprocesador matemático instalado
	0	Disco instalado
00412h	Reservado	
00413H-00414H	Tamaño de la memoria en K bytes (0-640K)	
00415H-00416H	Reservado	
00417H	Byte para control del teclado	
	Bit	Propósito
	7	Insertar bloqueado
	6	Mayúsculas bloqueada
	5	Números bloqueado
	4	Desplazamiento bloqueado
	3	Tecla alterna oprimida
	2	Tecla de control oprimida
	1	Tecla shift izquierda oprimida
	0	Tecla shift derecha oprimida
00418H	Byte para el control del teclado	
	Bit	Propósito
	7	Tecla para insertar oprimida
	6	Tecla de mayúsculas bloqueada oprimida
	5	Tecla de números bloqueada oprimida
	4	Tecla de desplazamiento bloqueada oprimida
	3	Pausa bloqueada
	2	Tecla de requerimiento del sistema oprimida
	1	Tecla alterna izquierda oprimida

TABLA A-11 *(Continuación)*

Localidad	*Propósito*
0	Tecla de control derecho oprimida
00419H	Entrada alterna al teclado
0041AH-0041BH	Apuntador del encabezado del búfer para teclado
0041CH-0041DH	Apuntador para fin del búfer para teclado
0041EH-0043DH	Area del búfer para teclado de 32 bytes
0043EH-00448H	Area de control de la unidad de disco
00449H-00466H	Area de información del control de video
00467H-0046BH	Reservado
0046CH-0046FH	Contador del medidor de tiempo
00470H	Desbordamiento del medidor de tiempo
00471H	Estado de la tecla de break
00472H-00473H	Reiniciar bandera
00474H-00477H	Area de información del lector del disco duro
00478H-0047BH	Area de tiempo fuera de LPT1-LPT4
0047CH-0047FH	Area de tiempo fuera de COM1-COM2
00480H-00481H	Apuntador inicial para el búfer desplazado del teclado
00482H-00483H	Apuntador final para el búfer desplazado del teclado
00484H-0048AH	Area de información del control de video
0048BH-00495H	Area de control de la unidad de disco duro
00496H	Modo de teclado, estado y tipo de bandera
00497H	Banderas LED del teclado
00498H-00499H	Dirección desplazada de bandera espera completa de usuario
0049AH-0049BH	Dirección del segmento de bandera espera completa de usuario
0049CH-0049DH	Cuenta de espera de usuario (palabra inferior)
0049EH-0049FH	Cuenta de espera de usuario (palabra superior)
004A0H	Bandera activa de espera
004A1H-004A7H	Reservado
004A8H-004ABH	Apuntador a los parámetros de video
004ACH-004EFH	Reservado
004F0H-004FFH	Area de comunicaciones del programa de aplicaciones
00500H	Imprimir el estado de la pantalla
00501H-00503H	Reservado
00504H	Estado del modo de unidad de disco sencillo
00505H-0050FH	Reservado
00510H-00521H	Utilizado por ROM BASIC
00522H-0052FH	Utilizado por DOS para la iniciación del disco
00530H-00533H	Utilizado por el comando MODE
00534H-005FFH	Reservado

EMM386.SYS I=E000=EFFF. Utilizando estos controladores incrementa al DOS TPA a más de 600K bytes. Un archivo CONFIG.SYS típico para DOS 5.0 aparece en el Ejemplo A-7. Observe que los controladores después de EMM386.SYS están cargados en la memoria alta con el directivo DEVICEHIGH en vez del directivo DEVICE. Los programas son cargados usando el directivo LOADHIGH o LH antes del nombre del programa.

EJEMPLO A–7

```
(Archivo CONFIG.SYS)

FILES=30
BUFFERS=30
STACKS=64,128
FCBS=48
SHELL=C:\DOS\COMMAND.COM C:\DOS\/E:256 /P
DEVICE=C:\DOS\HIMEM.SYS
DOS=HIGH,UMB
DEVICE=C:\DOS\EMM386.EXE I=C800-EFFF NOEMS
DEVICEHIGH SIZE=1EB0 C:\LASERLIB\SONY_CDU.SYS /D:SONY_001 /B:340 /Q:* /T:* /M:H
DEVICEHIGH SIZE=0910 C:\DOS\SETVER.EXE
DEVICEHIGH SIZE=3150 C:\MOUSE1\MOUSE.SYS
LASTDRIVE = F

(Archivo AUTOEXEC.BAT)

PATH C:\DOS;C:\;C:\MASM\BIN;C:\MASM\BINB\;C:\UTILITY;C:\WS;C:\LASERLIB
SET BLASTER=A220 I7 D1 T3
SET INCLUDE=C:\MASM\INCLUDE\
SET HELPFILES=C:\MASM\HELP\*.HLP
SET INIT=C:\MASM\INIT\
SET ASMEX=C:\MASM\SAMPLES\
SET TMP=C:\MASM\TMP
SET SOUND=C:\SB
LOADHIGH C:\LASERLIB\MSCDEX.EXE /D:SONY_001 /L:F /M:8
LOADHIGH C:\LASERLIB\LLTSR.EXE ALT-Q
LOADHIGH C:\DOS\FASTOPEN C:=256
LOADHIGH C:\DOS\DOSKEY /BUFSIZE=1024
LOADHIGH C:\LASERLIB\PRINTF.COM
DOSKEY GO=DOSSHELL
DOSSHELL
```

APENDICE B

Resumen del conjunto de instrucciones

El resumen del conjunto de instrucciones, que continúa después de esta introducción, contiene una lista completa de todas las instrucciones para los microprocesadores 8086, 8088, 80286, 80386 y 80486. Observe que las instrucciones para el coprocesador numérico del 80486 aparecen en el Capítulo 12.

Cada elemento de la instrucción mencionará el código neumónico de operación más una breve descripción del propósito de la instrucción. También se mencionan los códigos binarios en el lenguaje de máquina para cada instrucción más cualquier otra información requerida para formar la instrucción tal como el desplazamiento o la información inmediata. Al lado de la versión binaria de la instrucción en el lenguaje de máquina, aparecerán los bits del registro de la bandera así como cualquier cambio que pueda ocurrir para una instrucción específica. En esta lista, un espacio blanco indicará que no hay cambio, un ? indica un cambio con un resultado no predecible, un * indica un cambio predecible, un 1 indica que la bandera está establecida y un 0 indica que la bandera fue borrada.

Antes de empezar la lista de instrucciones, se requiere algo de información acerca de la configuración de los bits en las versiones binarias de las instrucciones en el lenguaje de máquina. La Tabla B-1 muestra los bits del modificador, codificados como 00 en las listas de las instrucciones, para que éstas puedan ser formadas con un registro, desplazadas o sin desplazar.

La Tabla B-2 menciona con el campo de registro/memoria los modos permitidos para direccionar la memoria, codificados como mmm. Esta tabla es aplicable a todas las versiones del microprocesador.

TABLA B–1 Los bits del modificador, codificados como 00 en la lista de instrucciones

oo	*Función*
00	Si mmm=110, entonces un desplazamiento sigue al código de operación; de otra manera, ningún desplazamiento es utilizado
01	Un desplazamiento identificado de 8-bits sigue al código de operación
10	Un desplazamiento identificado de 16-bits sigue al código de operación
11	mmm especifica un registro, en vez de a un modo de dirección

TABLA B–2 Descripción del campo de registro/memoria (mmm)

mmm	*Función*
000	DS:[BX+SI]
001	DS:[BX+DI]
010	SS:[BP+SI]
011	SS:[BP+DI]
100	DS:[SI]
101	DS:[DI]
110	SS:[BP]
111	DS:[BX]

TABLA B–3 Opciones del registro para campo (rrr)

rrr	*W=0*	*W=1*	*reg32*
000	AL	AX	EAX
001	CL	CX	ECX
010	DL	DX	EDX
011	BL	BX	EBX
100	AH	SP	ESP
101	CH	BP	EBP
110	DH	SI	ESI
111	BH	DI	EDI

La Tabla B-3 menciona las opciones del registro (rrr) cuando está codificado para un registro de 8-bits o 16-bits. Esta tabla también menciona los registros de 32-bits utilizados con los microprocesadores 80386 y 80486.

La Tabla B-4 menciona las asignaciones del bit del registro para segmento (rrr) para las instrucciones MOV, PUSH y POP, las cuales utilizan estos registros para los segmentos.

Cuando se utilizan los microprocesadores 80386 y 80486, algunas de las definiciones proporcionadas en las tablas anteriores cambian. Refiérase a las Tablas B-5 y B-6 para estos cambios y cómo se aplican a los microprocesadores 80386 y 80486.

El resumen del conjunto de instrucciones que sigue menciona todas las instrucciones, con ejemplos, para los microprocesadores 8086, 8088, 80286, 80386 y 80486. Faltan los prefijos para cambiar los segmentos: CS (2EH), SS (36H), DS (3EH), ES (26H), FS (64H) y GS (65H). Estos prefijos son de un byte de largo y colocados en la memoria antes de la instrucción que tiene el prefijo.

La Tabla B-7 muestra los cálculos de la dirección efectiva que se aplican sólo a los microprocesadores 8086 y 8088 como ea o a en el resumen del conjunto de instrucciones. Por ejemplo, la instrucción de ADD DATA,AL del 8086 16 + ea relojes. Debido a que la Tabla menciona que las direcciones desplazadas requieren 6 relojes, esta instrucción requiere de 22 relojes para ejecutar. Observe que todos los tiempos mencionados son los máximos y que en algunos casos el microprocesador puede ejecutar la instrucción en menos tiempo.

TABLA B–4 Asignaciones del registro para campo (rrr) que son usadas para representar los registros para segmentos

rrr	*Registro*
000	*ES*
001	*CS*
010	*SS*
011	*DS*

TABLA B–5 Los registros para índice están especificados con rrr en el microprocesador 80386 y 80486

rrr	*Registro para índice*
000	EAX
001	ECX
010	EDX
011	EBX
100	No índice
101	EBP
110	ESI
111	EDI

TABLA B–6 Las combinaciones posibles de 00, mmm, y rrr para el conjunto de instrucciones del 80386 y 80486 utilizando el modo de direccionamiento de 32-bits

oo	*mmm*	*rrr*	*Función*
00	000	—	DS:[EAX]
00	001	—	DS:[ECX]
00	010	—	DS:[EDX]
00	011	—	DS:[EBX]
00	100	000	DS:[EAX+índice progresiva]
00	100	001	DS:[ECX+índice progresiva]
00	100	010	DS:[EDX+índice progresiva]
00	100	011	DS:[EBX+índice progresiva]
00	100	100	SS:[ESP+índice progresiva]
00	100	101	DS:[disp32+índice progresiva]
00	100	110	DS:[ESI+índice progresiva]
00	100	111	DS:[EDI+índice progresiva]
00	101	—	DS:disp32
00	110	—	DS:[ESI]
00	111	—	DS:[EDI]
01	000	—	DS:[EAX+disp8]

TABLA B–6 *continuación*

oo	*mmm*	*rrr*	*Función*
01	001	—	DS:[ECX+disp8]
01	010	—	DS:[EDX+disp8]
01	011	—	DS:[EBX+disp8]
01	100	000	DS:[EAX+índice progresiva+disp8]
01	100	001	DS:[ECX+índice progresiva+disp8]
01	100	010	DS:[EDX+índice progresiva+disp8]
01	100	011	DS:[EBX+índice progresiva+disp8]
01	100	100	SS:[ESP+índice progresiva+disp8]
01	100	101	SS:[EBP+índice progresiva+disp8]
01	100	110	DS:[ESI+índice progresiva+disp8]
01	100	111	DS:[EDI+índice progresiva+disp8]
01	101	—	SS:[EBP+disp8]
01	110	—	DS:[ESI+disp8]
01	111	—	DS:[EDI+disp8]
10	000	—	DS:[EAX+disp32]
10	001	—	DS[ECX+disp32]
10	010	—	DS:[EDX+disp32]
10	011	—	DS:[EBX+disp32]
10	100	000	DS:[EAX+índice progresiva+disp32]
10	100	001	DS:[ECX+índice progresiva+disp32]
10	100	010	DS:[EDX+índice progresiva+disp32]
10	100	011	DS:[EBX+índice progresiva+disp32]
10	100	100	SS:[ESP+índice progresiva+disp32]
10	100	101	SS:[EBP+índice progresiva+disp32]
10	100	110	DS:[ESI+índice progresiva+disp32]
10	100	111	DS:[EDI+índice progresiva+disp32]
01	101	—	SS:[EBP+disp32]
01	110	—	DS:[ESI+disp32]
01	111	—	DS:[EDI+disp32]

Notas: disp8 = desplazamiento de 8 bits, disp32 = desplazamiento de 32 bits.

El bit-D, en el descriptor del segmento del código, indica el tamaño predeterminado del operando y la dirección para los microprocesadores 80386 y 80486. Si D = 1, entonces todas las direcciones y operandos son de 32 bits y si D = 0, todas las direcciones y operandos son de 16 bits. En el modo real, el bit-D será establecido en cero por los microprocesadores 80386 y 80486, así que los operandos y las direcciones son de 16 bits.

El prefijo del tamaño de la dirección (67H) se debe colocar antes de las instrucciones en el 80386 y 80486 para cambiar el tamaño predeterminado como fue seleccionado por el bit-D. Por

TABLA B–7 Cálculos de dirección efectivos para los microprocesadores 8086 y 8088

Tipo	*Relojes*	*Ejemplo*
Base o índice	5	MOV CL,[DI]
Desplazamiento	3	MOV AL,DATA
Base más índice [BP+DI] o [BX+SI]	7	MOV BL,[BP+DI]
Base más índice [BP+SI] o [BX+DI]	8	MOV CL,[BP+SI]
Desplazamiento más base o índice	9	MOV DH,[DI+20H]
Base más índice más desplazamiento [BP+DI+disp] o [BX+SI+disp]	11	MOV CX,DATA[BX+SI]
Base más índice más desplazamiento [BP+SI+disp] o [BX+DI+disp]	12	MOV CX,[BX+DI+2]
Prefijo para evitar el segmento	ea+2	MOV AL,ES:DATA

ejemplo, la instrucción MOV AX,[ECX] debe tener el prefijo del tamaño de la dirección colocado antes en el código de la máquina si el tamaño predeterminado es de 16 bits. Si el tamaño predeterminado es de 32 bits, el prefijo de la dirección no es necesario con esta instrucción. El prefijo para modificar operando (66H) funciona de la misma manera que el prefijo para el tamaño de la dirección. En el ejemplo anterior, el tamaño del operando es de 16-bits. Si el bit-D selecciona operandos y direcciones de 32-bits, esta instrucción requiere al prefijo para el tamaño del operando.

RESUMEN DEL CONJUNTO DE INSTRUCCIONES

AAA Ajuste de ASCII después de sumar

00110111

O	D	I	T	S	Z	A	P	C
?				?	?	*	?	*

Ejemplo		Relojes
AAA	8086	8
	8088	8
	80286	3
	80386	4
	80486	3

AAD Ajuste de ASCII antes de dividir

11010101 00001010

O	D	I	T	S	Z	A	P	C
?				*	*	?	*	?

Ejemplo Relojes

AAD	8086	60
	8088	60
	80286	14
	80386	19
	80486	14

AAM Ajuste de ASCII después de multiplicar

11010100 00001010

O D I T S Z A P C
? * * ? * ?

Ejemplo	Relojes	
AAM	8086	83
	8088	83
	80286	16
	80386	17
	80486	15

AAS Ajuste de ASCII después de restar

00111111

O D I T S Z A P C
? ? ? * ? *

Ejemplo	Relojes	
AAS	8086	8
	8088	8
	80286	3
	80386	4
	80486	3

ADC Sumar con restante

000100dw oorrrmmm disp

O D I T S Z A P C
* * * * * *

Formato	Ejemplos	Relojes	
ADC reg,reg	ADC AX,BX ADC AL,BL ADC EAX,EBX ADC CX,SI ADC ESI,EDI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1

Formato	Ejemplos		Relojes
ADC mem,reg	ADC DATA,AL ADC LIST,SI ADC DATA [DI],CL ADC [EAX],BL ADC [EBX+2*ECX],EDX	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	3
ADC reg,mem	ADC BL,DATA ADC SI,LIST ADC CL,DATA [DI] ADC CL,[EAX] ADC EDX,[EBX+100H]	8086	9+ea
		8088	13+ea
		80286	7
		80386	6
		80486	2

100000sw oo010mmm disp información

Formato	Ejemplos		Relojes
ADC reg,imm	ADC CX,3 ADC DI,1AH ADC DL,34H ADC EAX,12345 ADC CX,1234H	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
ADC mem,imm	ADC DATA,33 ADC LIST,'A' ADC DATA [DI],2 ADC BYTE PTR [EAX],3 ADC WORD PTR [DI],669H	8086	17+ea
		8088	23+ea
		80286	7
		80386	7
		80486	3

0001010w información

Formato	Ejemplos		Relojes
ADC acc,imm	ADC AX,3 ADC AL,1AH ADC AH,34 ADC EAX,3 ADC AL,'Z'	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

ADD Sumar

000000dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
ADD reg,reg	ADD AX,BX ADD AL,BL ADD EAX,EBX ADD CX,SI ADD ESI,EDI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
ADD mem,reg	ADD DATA,AL ADD LIST,SI ADD DATA [DI],CL ADD [EAX],CL ADD [EBX+4*EDX],EBX	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	3
ADD reg,mem	ADD BL,DATA ADD SI,LIST ADD CL,DATA [DI] ADD CL,[EAX] ADD EDX,[EBX+200H]	80286	9+ea
		8088	13+ea
		80286	7
		80386	6
		80486	2

100000sw oo000mmm disp información

Formato	Ejemplos	Relojes	
ADD reg,imm	ADD CX,3 ADD DI,1AH ADD DL,34H ADD EAX,123456 ADD CX,18AFH	80286	4
		8088	4
		80286	3
		80386	2
		80486	1
ADD mem,imm	ADD DATA,33 ADD LIST,'A' ADD DATA [DI],2 ADD BYTE PTR [EAX],3 ADD WORD PTR [DI],6A8H	8086	17+ea
		8088	23+ea
		80286	7
		80386	7
		80486	3

0000010w información

Formato	Ejemplos	Relojes	
ADD acc,imm	ADD AX,3 ADD AL,1AH ADD AH,56 ADD EAX,3 ADD AL,'D'	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

AND

AND lógico

001000dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Formato	Ejemplos	Relojes	
AND reg,reg	AND CX,BX AND DL,BL AND ECX,EBX AND BP,SI AND EDX,EDI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
AND mem,reg	AND BIT,CH AND LIST,DI AND DATA [BX],CL AND [ECX],AL AND [EDX+8*ECX],EDI	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	3
AND reg,mem	AND BL,DATA AND SI,LIST AND CL,DATA [DI] AND CL,[EAX] AND EDX,[EBX+34AH]	8086	9+ea
		8088	13+ea
		80286	7
		80386	6
		80486	2

100000sw oo100mmm disp información

Formato	Ejemplos	Relojes	
AND reg,imm	AND BP,1 AND DI,10H AND DL,34H AND EBP,12345 AND SP,1234H	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

AND mem,imm	AND DATA,33 AND LIST,4 AND DATA [SI],2 AND BYTE PTR [EAX],3 AND DWORD PTR [DI],32	8086	17+ea
		8088	23+ea
		80286	7
		80386	7
		80486	3

0010010w información

Formato	Ejemplos		Relojes
AND acc,imm	AND AX,15 AND AL,1FH AND AH,34 AND EAX,3 AND AL,'R'	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

ARPL Ajustar el nivel de privilegio requerido

01100011 oorrrmmm disp

O	D	I	T	S	Z	A	P	C
					*			

Formato	Ejemplos		Relojes
ARPL reg,reg	ARPL AX,BX ARPL BX,SI ARPL CX,DX ARPL BX,AX ARPL DI,SI	8086	—
		8088	—
		80286	10
		80386	20
		80486	9
ARPL mem,reg	ARPL NUMB,AX ARPL LIST,DI ARPL DATA [BX],CX ARPL [ECX],AX ARPL [EDX+4*ECX],DI	8086	—
		8088	—
		80286	11
		80386	21
		80486	9

BOUND Revisar límites de tabla

01100010 oorrrmmm disp

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos		Relojes
BOUND	BOUND AX,BETS	8086	—

reg,mem	BOUND BX,SAID BOUND CX,DATA BOUND BX,[DI] BOUND DI,[BX+2]	8088	—
		80286	13
		80386	10
		80486	7

BSF Verificación delantera de bit

00001111 10111100 oorrmmm disp

O D I T S Z A P C
(Z: *)

Formato	Ejemplos	Relojes	
BSF reg,reg	BSF AX,BX BSF BX,SI BSF ECX,EBX BSF EBX,EAX BSF DI,SI	8086	—
		8088	—
		80286	—
		80386	10+3n
		80486	6—42
BSF reg,mem	BSF AX,DATA BSF BP,LISTG BSF ECX,MEMORY BSF EAX,DATA6 BSF DI,[ECX]	8086	—
		8088	—
		80286	—
		80386	10+3n
		80486	7—43

BSR Verificación inversa de bit

00001111 10111101 oorrmmm disp

O D I T S Z A P C
(Z: *)

Formato	Ejemplos	Relojes	
BSR reg,reg	BSR AX,BX BSR BX,SI BSR ECX,EBX BSR EBX,EAX BSR DI,SP	8086	—
		8088	—
		80286	—
		80386	10+3n
		80486	6—103
BSR reg,mem	BSR AX,DATA BSR BP,LISTG BSR ECX,MEMORY BSR EAX,DATA6 BSR DI,[EBX]	8086	—
		8088	—
		80286	—
		80386	10+3n
		80486	7—104

BSWAP Intercambio de bytes

00001111 11001rrr

O D I T S Z A P C

Formato	Ejemplos	Relojes	
BSWAP reg	BSWAP EAX BSWAP EBX BSWAP ECX BSWAP EDX BSWAP EDI	8086	—
		8088	—
		80286	—
		80386	—
		80486	1

BT Prueba de bits

00001111 10111010 oo100mmm disp información

O D I T S Z A P C
*

Formato	Ejemplos	Relojes	
BT reg,imm8	BT AX,2 BT CX,4 BT BP,10H BT CX,8 BT BX,2	8086	—
		8088	—
		80286	—
		80386	3
		80486	3
BT mem,imm8	BT DATA1,2 BT LIST,2 BT DATA [DI],2 BT [BX],1 BT FROG,3	8086	—
		8088	—
		80286	—
		80386	6
		80486	3

00001111 10100011 disp

Formato	Ejemplos	Relojes	
BT reg,reg	BT AX,CX BT CX,DX BT BP,AX BT SI,CX BT CX,BP	8086	—
		8088	—
		80286	—
		80386	3
		80486	3

BT mem,reg	BT DATA1,AX BT LIST,DX BT DATA3,CX BT DATA9,BX BT DATA[[DI],AX	8086	—
		8088	—
		80286	—
		80386	12
		80486	8

BTC Prueba de bits y complemento

00001111 10111010 oo111mmm disp información — O D I T S Z A P C (C: *)

Formato	Ejemplos	Relojes	
BTC reg,imm8	BTC AX,2 BTC CX,4 BTC BP,10H BTC CX,8 BTC BX,2	8086	—
		8088	—
		80286	—
		80386	6
		80486	6
BTC mem,imm8	DATA1,2 BTC LIST,2 BTC DATA [DI],3 BTC [BX],1 BTC TOAD,5	8086	—
		8088	—
		80286	—
		80386	8
		80486	8

00001111 10111011 disp

Formato	Ejemplos	Relojes	
BTC reg,reg	BTC AX,CX BTC CX,DX BTC BP,AX BTC SI,CX BTC CX,BX	8086	—
		8088	—
		80286	—
		80386	6
		80486	6
BTC mem,reg	BTC DATA1,AX BTC LIST,DX BTC DATA3,CX BTC DATA9,BX BTC DATA [DI],AX	8086	—
		8088	—
		80286	—
		80386	13
		80486	13

BTR Prueba de bits y arranque nuevo

00001111 10111010 oo110mmm disp información — O D I T S Z A P C (C: *)

Formato	Ejemplos	Relojes	
BTR reg,imm8	BTR AX,2 BTR CX,4 BTR BP,10H BTR CX,8 BTR BX,2	8086	—
		8088	—
		80286	—
		80386	6
		80486	6
BTR mem,imm8	BTR DATA1,2 BTR LIST,2 BTR DATA [DI],4 BTR [BX],1 BTR SLED,6	8086	—
		8088	—
		80286	—
		80386	8
		80486	8

00001111 10110011 disp

Formato	Ejemplos	Relojes	
BTR reg,reg	BTR AX,CX BTR CX,DX BTR BP,AX BTR SI,CX BTR BP,CX	8086	—
		8088	—
		80286	—
		80386	6
		80486	6
BTR mem,reg	BTR DATA1,AX BTR LIST,DX BTR DATA3,CX BTR DATA9,BX BTR DATA [BX],AX	8086	—
		8088	—
		80286	—
		80386	13
		80486	13

BTS Prueba de bits y establecer

00001111 10111010 oo101mmm disp información — O D I T S Z A P C (C: *)

Formato	Ejemplos	Relojes	
BTS reg,imm8	BTS AX,2 BTS CX,4 BTS BP,10H BTS CX,8 BTS BX,3	8086	—
		8088	—
		80286	—
		80386	6
		80486	6

Formato	Ejemplos	Relojes	
BTS mem,imm8	BTS DATA1,2 BTS LIST,2 BTS DATA [BP],7 BTS [BX],1 BTS FROG,3	8086	—
		8088	—
		80286	—
		80386	8
		80486	8

00001111 10101011 disp

Formato	Ejemplos	Relojes	
BTS reg,reg	BTS AX,CX BTS CX,DX BTS BP,AX BTS SI,CX BTS CX,BP	8086	—
		8088	—
		80286	—
		80386	6
		80486	6
BTS mem,reg	BTS DATA1,AX BTS LIST,DX BTS DATA3,CX BTS DATA9,BX BTS DATA [BP],AX	8086	—
		8088	—
		80286	—
		80386	13
		80486	13

CALL

Procedimiento de llamar otro programa (subrutina)

11101000 disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
CALL label (near)	CALL FOR_FUN CALL HOME CALL ET CALL WAITING CALL SOMEONE	8086	19
		8088	23
		80286	7
		80386	3
		80486	3

10011010 disp

Formato	Ejemplos	Relojes	
CALL label (far)	CALL FAR PTR DATES CALL WHAT CALL WHERE CALL FARCE CALL WHOM	8086	28
		8088	36
		80286	13
		80386	17
		80486	18

11111111 oo010mmm

Formato	Ejemplos	Relojes	
CALL reg (near)	CALL AX CALL BX CALL CX CALL DI CALL SI	8086	16
		8088	20
		80286	7
		80386	7
		80486	5
CALL mem (near)	CALL ADDRESS CALL NEAR PTR [DI] CALL DATA CALL FROG CALL HERO	8086	21+ea
		8088	29+ea
		80286	11
		80386	10
		80486	5

11111111 oo011mmm

Formato	Ejemplos	Relojes	
CALL mem (far)	CALL FAR LIST [SI] CALL FROM HERE CALL TO THERE CALL SIXX CALL OCT	8086	37+ea
		8088	53+ea
		80286	16
		80386	22
		80486	17

CBW Convertir byte a palabra

10011000 O D I T S Z A P C

Ejemplo	Relojes	
CBW	8086	2
	8088	2
	80286	2
	80386	3
	80486	3

CDQ Convertir doble palabra a cuádruple palabra

10011001	O D I T S Z A P C	
Ejemplo	Relojes	
CDQ	8086	—
	8088	—
	80286	—
	80386	2
	80486	2

CLC Borrar bandera de restante

11111000	O D I T S Z A P C C = 0	
Ejemplo	Relojes	
CLC	8086	2
	8088	2
	80286	2
	80386	2
	80486	2

CLD Borrar bandera de dirección

11111100	O D I T S Z A P C D = 0	
Ejemplo	Relojes	
CLD	8086	2
	8088	2
	80286	2
	80386	2
	80486	2

CLI Borrar bandera de interrupción

11111010	O D I T S Z A P C I = 0	
Ejemplo	Relojes	
CLI	8086	2
	8088	2
	80286	3
	80386	3
	80486	5

CLTS — Borrar bandera de conmutador de tareas

00001111 00000110

O D I T S Z A P C

Ejemplo		Relojes
CLTS	8086	—
	8088	—
	80286	2
	80386	5
	80486	7

CMC — Complementar bandera de restante

10011000

O D I T S Z A P C (C: *)

Ejemplo		Relojes
CMC	8086	2
	8088	2
	80286	2
	80386	2
	80486	2

CMP — Comparar operandos

001110dw oorrrmmm disp

O D I T S Z A P C (O, S, Z, A, P, C: *)

Formato	Ejemplos		Relojes
CMP reg,reg	CMP AX,BX CMP AL,BL CMP EAX,EBX CMP CX,SI CMP ESI,EDI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
CMP mem,reg	CMP DATA,AL CMP LIST,SI CMP DATA [BX],CL CMP [EAX],AL CMP [EBX+2*ECX],EBX	8086	9+ea
		8088	13+ea
		80286	7
		80386	5
		80486	2

CMP reg,mem	CMP BL,DATA CMP SI,LIST CMP CL,DATA [DI] CMP CL,[EAX] CMP EDX,[EBX+200H]	8086	9+ea
		8088	13+ea
		80286	6
		80386	6
		80486	2

100000sw oo111mmm disp información

Formato	Ejemplos		Relojes
CMP reg,imm	CMP CX,3 CMP DI,1AH CMP DL,34H CMP EBX,12345 CMP CX,123AH	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
CMP mem,imm	CMP DATA,33 CMP LIST,'A' CMP DATA [DI],87H CMP BYTE PTR [EAX],3 CMP WORD PTR [SI],7	8086	10+ea
		8088	14+ea
		80286	6
		80386	5
		80486	2

0011110w información

Formato	Ejemplos		Relojes
CMP acc,imm	CMP AX,3 CMP AL,1AH CMP AH,34 CMP EAX,3 CMP AL,'I'	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

CMPS Comparar cadenas

1010011w

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos		Relojes
CMPSB CMPSW CMPSD	CMPSB CMPSW CMPSD CMPS DATA1 REPE CMPSB REPNE CMPSW	8086	22
		8088	30
		80286	8
		80386	10
		80486	8

CMPXCHG Comparar e intercambiar

00001111 1011000w 11rrrrrr

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
CMPXCHG reg,reg	CMPXCHG EAX,EBX CMPXCHG ECX,EDX	8086	—
		8088	—
		80286	—
		80386	—
		80486	6

00001111 1011000w oorrrmmm

Formato	Ejemplos	Relojes	
CMPXCHG mem,reg	CMPXCHG DATA,EAX CMPXCHG DATA2,EBX	8086	—
		8088	—
		80286	—
		80386	—
		80486	7

CWD Convertir palabra a doble palabra

10011001

O	D	I	T	S	Z	A	P	C

Ejemplo	Relojes	
CWD	8086	5
	8088	5
	80286	2
	80386	2
	80486	3

CWDE Convertir palabra a doble palabra extendida

10011000

O	D	I	T	S	Z	A	P	C

Ejemplo	Relojes	
CWDE	8086	—
	8088	—
	80286	—
	80386	3
	80486	3

DAA — Ajuste de decimales después de sumar

00100111

O	D	I	T	S	Z	A	P	C
?				*	*	*	*	*

Ejemplo		Relojes
DAA	8086	4
	8088	4
	80286	3
	80386	4
	80486	2

DAS — Ajuste de decimales después de restar

00101111

O	D	I	T	S	Z	A	P	C
?				*	*	*	*	*

Ejemplo		Relojes
DAS	8086	4
	8088	4
	80286	3
	80386	4
	80486	2

DEC — Decremento

1111111w oo001mmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	

Formato	Ejemplos		Relojes
DEC reg8	DEC BL DEC BH DEC CL DEC DH DEC AH	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
DEC mem	DEC DATA DEC LIST DEC DATA [SI] DEC BYTE PTR [EAX] DEC WORD PTR [DI]	8086	15+ea
		8088	23+ea
		80286	7
		80386	6
		80486	3

01001rrr

Formato	Ejemplos	Relojes	
DEC reg16 DEC reg32	DEC AX DEC EAX DEC CX DEC EBX DEC DI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1

DIV División sin identificación

1111011w oo110mmm disp

O D I T S Z A P C
? ? ? ? ? ?

Formato	Ejemplos	Relojes	
DIV reg	DIV BL DIV BH DIV ECX DIV BH DIV CH	8086	162
		8088	162
		80286	22
		80386	38
		80486	40
DIV mem	DIV DATA DIV LIST DIV DATA [DI] DIV BYTE PTR [EAX] DIV WORD PTR [DI]	8086	168
		8088	176
		80286	25
		80386	41
		80486	40

ENTER Crear un marco de pila

11001000 información

O D I T S Z A P C

Formato	Ejemplos	Relojes	
ENTER imm,0	ENTER 4,0 ENTER 8,0 ENTER 100,0 ENTER 200,0 ENTER 1024,0	8086	—
		8088	—
		80286	11
		80386	10
		80486	14

ENTER imm,1	ENTER 4,1 ENTER 10,1	8086	—
		8088	—
		80286	15
		80386	12
		80486	17
ENTER imm,imm	ENTER 3,6 ENTER 100,3	8086	—
		8088	—
		80286	12
		80386	15
		80486	17

ESC Escape

11011nnn oonnnmmm — O D I T S Z A P C

nnnnnn = Código de operación para coprocesador

Formato	Ejemplos	Relojes	
ESC imm,reg	ESC 5,AL ESC 5,BH ESC 6,CH FADD ST,ST(3)	8086	2
		8088	2
		80286	20
		80386	var
		80486	var
ESC imm,mem	ESC 2,DATA ESC 3,FROG FADD DATA FMUL FROG	8086	8+ea
		8088	12+ea
		80286	20
		80386	var
		80486	var

HLT Alto

11110100 — O D I T S Z A P C

Ejemplo	Relojes	
HLT	8086	2
	8088	2
	80286	2
	80386	5
	80486	4

IDIV División con identificación

1111011w oo111mmm disp

O	D	I	T	S	Z	A	P	C
?				?	?	?	?	?

Formato	Ejemplos		Relojes
IDIV reg	IDIV BL IDIV BH IDIV ECX IDIV BH IDIV CX	8086	184
		8088	184
		80286	25
		80386	43
		80486	43
IDIV mem	IDIV DATA IDIV LIST IDIV DATA [DI] IDIV BYTE PTR [EAX] IDIV WORD PTR [DI]	8086	190
		8088	194
		80286	28
		80386	46
		80486	44

IMUL Multiplicación con identificación

1111011w oo101mmm disp

O	D	I	T	S	Z	A	P	C
*				?	?	?	?	*

Formato	Ejemplos		Relojes
IMUL reg	IMUL BL IMUL CL IMUL CX IMUL ECX IMUL EBX	8086	154
		8088	154
		80286	21
		80386	38
		80486	42
IMUL mem	IMUL DATA IMUL LIST IMUL DATA [SI] IMUL BYTE PTR [EAX] IMUL WORD PTR [DI]	8086	160
		8088	164
		80286	24
		80386	41
		80486	42

011010sl oorrrmmm disp información

Formato	Ejemplos	Relojes	
IMUL reg,imm	IMUL CX,16 IMUL DX,100 IMUL EAX,20	8086	—
		8088	—
		80286	21
		80386	38
		80486	42
IMUL reg,reg,imm	IMUL DX,AX,2 IMUL CX,DX,3 IMUL BX,AX,33	8086	—
		8088	—
		80286	21
		80386	38
		80486	42
IMUL reg,mem,imm	IMUL CX,DATA,4	8086	—
		8088	—
		80286	24
		80386	38
		80486	42

00001111 10101111 oorrrmmm disp

Formato	Ejemplos	Relojes	
IMUL reg,reg	IMUL CX,DX IMUL DX,BX IMUL EAX,ECX	8086	—
		8088	—
		80286	—
		80386	38
		80486	42
IMUL reg,mem	IMUL DX,DATA IMUL CX,FROG IMUL BX,LISTS	8086	—
		8088	—
		80286	—
		80386	41
		80486	42

IN

Entrada de información del puerto

1110010w número del puerto — O D I T S Z A P C

Formato	Ejemplos	Relojes

IN acc,pt	IN AL,12H IN AX,12H IN AL,0FFH IN AX,0FFH IN EAX,10H	8086	10
		8088	14
		80286	5
		80386	12
		80486	14

1110110w

Formato	Ejemplos		Relojes
IN acc,DX	IN AL,DX IN AX,DX IN EAX,DX	8086	8
		8088	12
		80286	5
		80386	13
		80486	14

INC - Incremento

1111111w oo000mmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	

Formato	Ejemplos		Relojes
INC reg8	INC BL INC BH INC CL INC DH INC AH	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
INC mem	INC DATA INC LIST INC DATA [BX] INC BYTE PTR [EAX] INC WORD PTR [BX] INC DWORD PTR [ECX]	8086	15+ea
		8088	23+ea
		80286	7
		80386	6
		80486	3

01000rrr

Formato	Ejemplos		Relojes
INC reg16 INC reg32	INC AX INC EAX INC CX INC EBX INC DI	8086	3
		8088	3
		80286	2
		80386	2
		80486	1

INS Cadena de entrada de información del puerto

0110110w — O D I T S Z A P C

Formato	Ejemplos	Relojes	
INSB INSW INSD	INSB INSW INSD INS DATA REP INSB	8086	—
		8088	—
		80286	5
		80386	15
		80486	17

INT Interruptor

11001101 tipo — O D I T S Z A P C

Formato	Ejemplos	Relojes	
INT type	INT 10H INT 255 INT 21H INT 20H INT 15H	8086	51
		8088	71
		80286	23
		80386	37
		80486	30

11001100

Ejemplo	Relojes	
INT 3	8086	52
	8088	72
	80286	23
	80386	33
	80486	26

INTO Interruptor en desbordamiento

11001110 — O D I T S Z A P C

Ejemplo	Relojes	
INTO	8086	53
	8088	73
	80286	24
	80386	35
	80486	28

INVD Invalidar al cache para información

00001111 00001000 — O D I T S Z A P C

Ejemplo		Relojes
INVD	8086	—
	8088	—
	80286	—
	80386	—
	80486	4

INVLPG Invalidar elemento TLB

00001111 00000001 oo111mmm — O D I T S Z A P C

Formato	Ejemplos		Relojes
INVLPG mem	INVLPG DATA INVLPG LIST	8086	—
		8088	—
		80286	—
		80386	—
		80486	12

IRET Retorno de interruptor

11001101 información

O	D	I	T	S	Z	A	P	C
*	*	*	*	*	*	*	*	*

Formato	Ejemplos		Relojes
IRET IRETD	IRET IRETD IRET 10H	8086	32
		8088	44
		80286	17
		80386	22
		80486	15

Jconditional Salto condicional

0111cccc disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
Jcc label (8-bit disp)	JA BELOW JB ABOVE JG GREATER JE EQUAL JZ ZERO	8086	16/4
		8088	16/4
		80286	7/3
		80386	7/3
		80486	3/1

00001111 1000cccc disp

Formato	Ejemplos	Relojes	
Jcc label (16-bit disp)	JNE NOT_MORE JLE LESS_THAN	8086	—
		8088	—
		80286	—
		80386	7/3
		80486	3/1

Condición Códigos	Mnemonic	Bandera	Descripción
0000	JO	O = 1	Saltar si existe desbordamiento
0001	JNO	O = 0	Saltar si no existe desbordamiento
0010	JB/JNAE	C = 1	Saltar si está abajo
0011	JAE/JNB	C = 0	Saltar si está arriba o igual
0100	JE/JZ	Z = 1	Saltar si es igual/cero
0101	JNE/JNZ	Z = 0	Saltar si no es igual/no es cero
0110	JBE/JNA	C = 1 + Z = 1	Saltar si está abajo o igual
0111	JA/JNBE	C = 0 • Z = 0	Saltar si está arriba
1000	JS	S = 1	Saltar si existe signo
1001	JNS	S = 0	Saltar si no existe signo
1010	JP/JPE	P = 1	Saltar si la paridad es pares
1011	JNP/JPO	P = 0	Saltar si la paridad es nones
1100	JL/JNGE	S • O	Saltar si es menor que
1101	JGE/JNL	S = O	Saltar si es mayor o igual
1110	JLE/JNG	Z = 1 + S • O	Saltar si es menor o igual
1111	JG/JNLE	Z = 0 + S = O	Saltar si es mayor

JCXZ/JECXZ Saltar si CX (ECX) es igual a cero

11100011

O D I T S Z A P C

Formato	Ejemplos		Relojes
JCXZ label JECXZ label	JCXZ LOTSA JCXZ OVER JECXZ UPPER JECXZ UNDER JCXZ NEXT	8086	18/6
		8088	18/6
		80286	8/4
		80386	9/5
		80486	8/5

JMP Salto incondicional

11101011 disp

O D I T S Z A P C

Formato	Ejemplos		Relojes
JMP label (corto)	JMP SHORT UP JMP SHORT DOWN JMP SHORT OVER JMP SHORT CIRCUIT JMP SHORT ARM	8086	15
		8088	15
		80286	7
		80386	7
		80486	3

11101001 disp

Formato	Ejemplos		Relojes
JMP label (cerca)	JMP VER JMP FROG JMP UNDER JMP NEAR PTR OVER	8086	15
		8088	15
		80286	7
		80386	7
		80486	3

11101010 disp

Formato	Ejemplos		Relojes
JMP label (lejos)	JMP VER JMP FROG JMP UNDER JMP FAR PTR THERE	8086	15
		8088	15
		80286	11
		80386	12
		80486	17

11111111 oo100mmm

Formato	Ejemplos	Relojes	
JMP reg (cerca)	JMP AX JMP EAX JMP CX JMP DX	8086	11
		8088	11
		80286	7
		80386	7
		80486	3
JMP mem (cerca)	JMP DATA JMP LIST JMP DATA [DI+2]	8086	18+ea
		8088	18+ea
		80286	11
		80386	10
		80486	5

11111111 oo101mmm

Formato	Ejemplos	Relojes	
JMP mem (lejos)	JMP WAY_OFF JMP TABLE JMP UP	8086	24+ea
		8088	24+ea
		80286	15
		80386	12
		80486	13

LAHF Cargar AH de las banderas

10011111 O D I T S Z A P C

Ejemplo	Relojes	
LAHF	8086	4
	8088	4
	80286	2
	80386	2
	80486	3

LAR — Cargar los derechos de acceso

00001111 00000010 oorrrmmm disp

O	D	I	T	S	Z	A	P	C
					*			

Formato	Ejemplos	Relojes	
LAR reg,reg	LAR AX,BX LAR CX,DX LAR EAX,ECX	8086	—
		8088	—
		80286	14
		80386	15
		80486	11
LAR reg,mem	LAR CX,DATA LAR AX,LIST LAR ECX,FROG	8086	—
		8088	—
		80286	16
		80386	16
		80486	11

LDS — Cargar apuntador lejano

11000101 oorrrmmm

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos	Relojes	
LDS reg,mem	LDS DI,DATA LDS SI,LIST LDS BX,ARRAY LDS CX,PNTR	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	6

LES — Cargar apuntador lejano

11000100 oorrrmmm

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos	Relojes	
LES reg,mem	LES DI,DATA LES SI,LIST LES BX,ARRAY LES CX,PNTR	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	6

LFS Cargar apuntador lejano

00001111 10110100 oorrrmmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LFS reg,mem	LFS DI,DATA LFS SI,LIST LFS BX,ARRAY LFS CX,PNTR	8086	—
		8088	—
		80286	—
		80386	7
		80486	6

LGS Cargar apuntador lejano

00001111 10110101 oorrrmmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LGS reg,mem	LGS DI,DATA LGS SI,LIST LGS BX,ARRAY LGS CX,PNTR	8086	—
		8088	—
		80286	—
		80386	7
		80486	6

LSS Cargar apuntador lejano

00001111 10110010 oorrrmmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LSS reg,mem	LSS DI,DATA LSS SI,LIST LSS BX,ARRAY LSS CX,PNTR	8086	—
		8088	—
		80286	—
		80386	7
		80486	6

LEA Cargar la dirección efectiva

10001101 oorrrmmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LEA reg,mem	LEA DI,DATA LEA SI,LIST LEA BX,ARRAY LEA CX,PNTR LEA BP,ADDR	8086	2+ea
		8088	2+ea
		80286	3
		80386	2
		80486	2

LEAVE — Salir del procedimiento de nivel alto

11001001 — O D I T S Z A P C

Ejemplo	Relojes	
LEAVE	8086	—
	8088	—
	80286	5
	80386	4
	80486	5

LGDT — Cargar tabla del descriptor global

00001111 00000001 oo010mmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LGDT mem64	LGDT DESCRIP LGDT TABLE	8086	—
		8088	—
		80286	11
		80386	11
		80486	11

LIDT — Cargar tabla del interruptor para descriptor

00001111 00000001 oo011mmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LIGT mem64	LIDT DATA LIDT DESCRIP	8086	—
		8088	—
		80286	12
		80386	11
		80486	11

LLDT Cargar tabla local del descriptor

00001111 00000000 oo010mmm disp — O D I T S Z A P C

Formato	Ejemplos	Relojes	
LLDT reg	LLDT AX LLDT CX	8086	—
		8088	—
		80286	17
		80386	20
		80486	11
LLDT mem	LLDT DATA LLDT LIST	8086	—
		8088	—
		80286	19
		80386	24
		80486	11

LMSW Cargar palabra del estado de la máquina

00001111 00000001 oo110mmm disp — O D I T S Z A P C

Solo se debe utilizar con el 80286

Formato	Ejemplos	Relojes	
LMSW reg	LMSW AX LMSW CX	8086	—
		8088	—
		80286	3
		80386	10
		80486	2
LMSW mem	LMSW DATA LMSW LIST	8086	—
		8088	—
		80286	6
		80386	13
		80486	3

LOCK Cerrar el canal

11110000		O D I T S Z A P C	
Formato	Ejemplos	Relojes	
LOCK inst	LOCK:XCHG AX,BX LOCK:MOV AL,AH	8086	2
		8088	2
		80286	0
		80386	0
		80486	1

LODS Cargar el operando de la cadena

1010110w		O D I T S Z A P C	
Formato	Ejemplos	Relojes	
LODSB LODSW LODSD	LODSB LODSW LODSD LODS DATA LODS ES:DATA	8086	12
		8088	16
		80286	5
		80386	5
		80486	5

LOOP Repetir hasta que CX = 0

11100010 disp		O D I T S Z A P C	
Formato	Ejemplos	Relojes	
LOOP label	LOOP DATA LOOP BACK	8086	17/5
		8088	17/5
		80286	8/4
		80386	11
		80486	7/6

LOOPE Repetir mientras sea igual

11100001 disp		O D I T S Z A P C	
Formato	Ejemplos	Relojes	
LOOPE label LOOPZ label	LOOPE NEXT LOOPE AGAIN LOOPZ REPEAT	8086	18/6
		8088	18/6
		80286	8/4
		80386	11
		80486	9/6

LOOPNE Repetir mientras que no sea igual

11100000 disp

O D I T S Z A P C

Formato	Ejemplos		Relojes
LOOPNE label LOOPNZ label	LOOPNE AGAIN LOOPNE BACK LOOPNZ REPL	8086	19/5
		8088	19/5
		80286	8/4
		80386	11
		80486	9/6

LSL Cargar el límite del segmento

00001111 00000011 oorrrmmm disp

O D I T S Z A P C (Z: *)

Formato	Ejemplos		Relojes
LSL reg,reg	LSL AX,BX LSL CX,BX LSL DX,AX	8086	—
		8088	—
		80286	14
		80386	25
		80486	10
LSL reg,mem	LSL AX,LIMIT LSL EAX,NUMB	8086	—
		8088	—
		80286	16
		80386	26
		80486	10

LTR Cargar registro para tareas

00001111 00000000 oo001mmm disp

O D I T S Z A P C

Formato	Ejemplos		Relojes
LTR reg	LTR AX LTR CX LTR DX	8086	—
		8088	
		80286	17
		80386	23
		80486	20

LTR mem	LTR TASK LTR EDGE	8086	—
		8088	—
		80286	19
		80386	27
		80486	20

MOV Mover información

100010dw oorrrmmm disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
MOV reg,reg	MOV CL,CH MOV BH,CL MOV CX,DX MOV EAX,ECX MOV EBP,ESI	8086	2
		8088	2
		80286	2
		80386	2
		80486	1
MOV mem,reg	MOV DATA,DL MOV NUMB,CX MOV TEMP,EBX MOV TEMP1,CH MOV DATA2,CL	8086	9+ea
		8088	13+ea
		80286	3
		80386	2
		80486	1
MOV reg,mem	MOV DL,DATA MOV DX,NUMB MOV EBX,TEMP MOV CH,TEMP1 MOV CL,DATA2	8086	10+ea
		8088	12+ea
		80286	5
		80386	4
		80486	1

1100011w oo000mmm disp información

Formato	Ejemplos	Relojes	
MOV mem,imm	MOV DATA,23H MOV LIST,12H MOV BYTE PTR [DI],2 MOV NUMB,234H MOV DWORD PTR [SI],100	8086	10+ea
		8088	14+ea
		80286	3
		80386	2
		80486	1

1011wrrr información

Formato	Ejemplos		Relojes
MOV reg,imm	MOV BX,23H MOV CX,12H MOV CL,2 MOV ECX,123423H MOV DI,100	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

101000dw disp

Formato	Ejemplos		Relojes
MOV mem,acc	MOV DATA,AL MOV NUMB,AX MOV NUMB1,EAX	8086	10
		8088	14
		80286	3
		80386	2
		80486	1
MOV acc,mem	MOV AL,DATA MOV AX,NUMB MOV EAX,TEMP	8086	10
		8088	14
		80286	5
		80386	4
		80486	1

100011d0 oosssmmm disp

Formato	Ejemplos		Relojes
MOV seg,reg	MOV SS,AX MOV DS,DX MOV ES,CX	8086	2
		8088	2
		80286	2
		80386	2
		80486	1
MOV seg,mem	MOV SS,DATA MOV DS,NUMB MOV ES,TEMP1	8086	8+ea
		8088	12+ea
		80286	2
		80386	2
		80486	1

MOV reg,seg	MOV AX,DS MOV DX,ES MOV CX,CS	8086	2
		8088	2
		80286	2
		80386	2
		80486	1
MOV mem,seg	MOV DATA,SS MOV NUMB,ES MOV TEMP1,DS	8086	9+ea
		8088	13+ea
		80286	3
		80386	2
		80486	1

00001111 001000d0 11rrrmmm

Formato	Ejemplos		Relojes
MOV reg,cr	MOV EAX,CR0 MOV EBX,CR2 MOV ECX,CR3	8086	—
		8088	—
		80286	—
		80386	6
		80486	4
MOV cr,reg	MOV CR0,EAX MOV CR2,EBX MOV CR3,ECX	8086	—
		8088	—
		80286	—
		80386	10
		80486	4

00001111 001000d1 11rrrmmm

Formato	Ejemplos		Relojes
MOV reg,dr	MOV EBX,DR6 MOV EAX,DR6 MOV EDX,DR1	8086	—
		8088	—
		80286	—
		80386	22
		80486	10

MOV dr,reg	MOV DR1,ECX MOV DR2,ESI MOV DR6,EBP	8086	—
		8088	—
		80286	—
		80386	22
		80486	11

00001111 001001d0 11rrrmmm

Formato	Ejemplos	Relojes	
MOV reg,tr	MOV EAX,TR6 MOV EDX,TR7	8086	—
		8088	—
		80286	—
		80386	12
		80486	4
MOV tr,seg	MOV TR6,EDX MOV TR7,ESI	8086	—
		8088	—
		80286	—
		80386	12
		80486	6

MOVS Mover información de la cadena

1010010w — O D I T S Z A P C

Formato	Ejemplos	Relojes	
MOVSB MOVSW MOVSD	MOVSB MOVSW MOVSD MOVS DAT1,DAT2 REP MOVSB	8086	18
		8088	26
		80286	5
		80386	7
		80486	7

MOVSX Moverse con signo extendido

00001111 1011111w oorrrmmm disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
MOVSX reg,reg	MOVSX BX,AL MOVSX EAX,DX	8086	—
		8088	—
		80286	—
		80386	3
		80486	3
MOVSX reg,mem	MOVSX AX,DATA MOVSX EAX,NUMB	8086	
		8088	—
		80286	—
		80386	6
		80486	3

MOVZX Moverse con cero extendido

00001111 1011011w oorrrmmm disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
MOVZX reg,reg	MOVZX BX,AL MOVZX EAX,DX	8086	—
		8088	—
		80286	—
		80386	3
		80486	3
MOVZX reg,mem	MOVZX AX,DATA MOVZX EAX,NUMB	8086	—
		8088	—
		80286	—
		80386	6
		80486	3

MUL Multiplicación sin identificar

1111011w oo100mmm disp

O	D	I	T	S	Z	A	P	C
*				?	?	?	?	*

Formato	Ejemplos	Relojes	
MUL reg	MUL BL MUL CX MUL ECX	8086	118
		8088	143
		80286	21
		80386	38
		80486	42
MUL mem	MUL DATA MUL BYTE PTR [SI] MUL WORD PTR [SI] MUL DWORD PTR [ECX]	8086	139
		8088	143
		80286	24
		80386	41
		80486	42

NEG Negación

1111011w oo011mmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
NEG reg	NEG AX NEG CX NEG EDX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
NEG mem	NEG DATA NEG NUMB NEG WORD PTR [DI]	8086	16+ea
		8088	24+ea
		80286	7
		80386	6
		80486	3

NOP Sin operación

10010000

O	D	I	T	S	Z	A	P	C

Ejemplo	Relojes	
NOP	8086	3
	8088	3
	80286	3
	80386	3
	80486	3

NOT El complemento de uno

1111011w oo010mmm disp

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos	Relojes	
NOT reg	NOT AX NOT CX NOT EDX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
NOT mem	NOT DATA NOT NUMB NOT WORD PTR [DI]	8086	16+ea
		8088	24+ea
		80286	7
		80386	6
		80486	3

OR OR-inclusivo

000010dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Formato	Ejemplos	Relojes	
OR reg,reg	OR CL,BL OR CX,DX OR ECX,EBX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1

OR mem,reg	OR DATA,CL OR NUMB,CX OR [DI],CX	8086	16+ea
		8088	24+ea
		80286	7
		80386	7
		80486	3
OR reg,mem	OR CL,DATA OR CX,NUMB OR CX,[SI]	8086	9+ea
		8088	13+ea
		80286	7
		80386	6
		80486	2

100000sw oo001mmm disp data

Formato	Ejemplos		Relojes
OR reg,imm	OR CL,3 OR DX,1000H OR EBX,100000H	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
OR mem,imm	OR DATA,33 OR NUMB,4AH OR NUMS,123498H OR BYTE PTR [ECX],2	8086	17+ea
		8088	25+ea
		80286	7
		80386	7
		80486	3

0000110w información

Formato	Ejemplos		Relojes
OR acc,imm	OR AL,3 OR AX,1000H OR EAX,100000H	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

OUT

Salida de información al puerto

1110011w número de puerto — O D I T S Z A P C

Formato	Ejemplos	Relojes	
OUT pt,acc	OUT 12H,AL OUT 12H,AX OUT 0FFH,AL OUT 0FEH,AX OUT 10H,EAX	8086	10
		8088	14
		80286	3
		80386	10
		80486	10

1110111w

Formato	Ejemplos	Relojes	
OUT DX,acc	OUT DX,AL OUT DX,AX OUT DX,EAX	8086	8
		8088	12
		80286	3
		80366	11
		80486	10

OUTS

Salida de información de la cadena al puerto

1110011w número de puerto — O D I T S Z A P C

Formato	Ejemplos	Relojes	
OUTSB OUTSW OUTSD	OUTSB OUTSW OUTSD OUTS DATA REP OUTSB	8086	—
		8088	—
		80286	5
		80386	14
		80486	10

POP

Sacar datos de la pila

01011rrr — O D I T S Z A P C

Formato	Ejemplos	Relojes	
POP reg	POP CX POP AX POP EBX	8086	8
		8088	12
		80286	5
		80386	4
		80486	1

10001111 oo000mmm disp

Formato	Ejemplos	Relojes	
POP mem	POP DATA POP LISTS POP NUMBS	8086	17+ea
		8088	25+ea
		80286	5
		80386	5
		80486	4

00sss111

Formato	Ejemplos	Relojes	
POP seg	POP DS POP ES POP SS	8086	8
		8088	12
		80286	5
		80386	7
		80486	3

00001111 10sss001

Formato	Ejemplos	Relojes	
POP seg	POP FS POP GS	8086	—
		8088	—
		80286	—
		80386	7
		80486	3

POPA/POPAD Sacar todos los registros de la pila

01100001 O D I T S Z A P C

Ejemplo	Relojes	
POPA POPAD	8086	—
	8088	—
	80286	19
	80386	24
	80486	9

POPF/POPFD Sacar las banderas de la pila

10011101

O	D	I	T	S	Z	A	P	C
*	*	*	*	*	*	*	*	*

Ejemplo		Relojes
POPF POPFD	8086	—
	8088	—
	80286	5
	80386	5
	80486	6

PUSH Empujar información a la pila

01010rrr

O D I T S Z A P C

Formato	Ejemplos		Relojes
PUSH reg	PUSH CX PUSH AX PUSH ECX	8086	11
		8088	15
		80286	3
		80386	2
		80486	1

11111111 oo110mmm disp

Formato	Ejemplos		Relojes
PUSH mem	PUSH DATA PUSH LISTS PUSH NUMB PUSH DWORD PTR [ECX]	8086	16+ea
		8088	24+ea
		80286	5
		80386	5
		80486	4

00sss110

Formato	Ejemplos		Relojes
PUSH seg	PUSH DS PUSH CS PUSH ES	8086	10
		8088	14
		80286	3
		80386	2
		80486	3

00001111 10sss000

Formato	Ejemplos	Relojes	
PUSH seg	PUSH FS PUSH GS	8086	—
		8088	—
		80286	—
		80386	2
		80486	3

011010s0 información

Formato	Ejemplos	Relojes	
PUSH imm	PUSH 2000H PUSH 5322H PUSHW 10H PUSHD 100000H	8086	—
		8088	—
		80286	3
		80386	2
		80486	1

PUSHA/PUSHAD Empujar a todos los registros

01100000 — O D I T S Z A P C

Ejemplo	Relojes	
PUSHA PUSHAD	8086	—
	8088	—
	80286	17
	80386	18
	80486	11

PUSHF/PUSHFD Empujar las banderas a la pila

10011100 — O D I T S Z A P C

Ejemplo	Relojes	
PUSHF PUSHFD	8086	10
	8088	14
	80286	3
	80386	4
	80486	3

RCL/RCR/ROL/ROR Rotar

1101000w ooTTTmmm disp

O D I T S Z A P C
* *

TTT = 000 = ROL
TTT = 001 = ROR
TTT = 010 = RCL
TTT = 011 = RCR

Formato	Ejemplos		Relojes
ROL reg,1 ROR reg,1	ROL CL,1 ROL DX,1 ROR CH,1 ROL SI,1	8086	2
		8088	2
		80286	2
		80386	3
		80486	3
RCL reg,1 RCR reg,1	RCL CL,1 RCL SI,1 RCR AH,1 RCR EBX,1	8086	2
		8088	2
		80286	2
		80386	9
		80486	3
ROL mem,1 ROR mem,1	ROL DATA,1 ROL BYTE PTR [DI],1 ROR NUMB,1 ROR DWORD PTR [ECX],1	8086	15+ea
		8088	23+ea
		80286	7
		80386	7
		80486	4
RCL mem,1 RCR mem,1	RCL DATA,1 RCL BYTE PTR [DI],1 RCR NUMB,1 RCR WORD PTR [ECX],1	8086	15+ea
		8088	23+ea
		80286	7
		80386	10
		80486	4

1101001w ooTTTmmm disp

Formato	Ejemplos		Relojes
ROL reg,CL ROR reg,CL	ROL CH,CL ROL DX,CL ROR CH,CL ROL SI,CL	8086	8+4n
		8088	8+4n
		80286	5+n
		80386	3
		80486	3

RCL reg,CL RCR reg,CL	RCL DL,CL RCL SI,CL RCR AH,CL RCR BX,CL	8086	8+4n
		8088	8+4n
		80286	5+n
		80386	9
		80486	8
ROL mem,CL ROR mem,CL	ROL DATA,CL ROL BYTE PTR [DI],CL ROR NUMB,CL ROR WORD PTR [ECX],CL	8086	20+a+4n
		8088	28+a+4n
		80286	8+n
		80386	7
		80486	4
RCL mem,CL RCR mem,CL	RCL DATA,CL RCL BYTE PTR [DI],CL RCR NUMB,CL RCR WORD PTR [ECX],CL	8086	20+a+4n
		8088	28+a+4n
		80286	8+n
		80386	10
		80486	9

1100000w ooTTTmmm disp información

Formato	Ejemplos		Relojes
ROL reg,imm ROR reg,imm	ROL CL,4 ROL DX,5 ROR CH,12 ROL SI,9	8086	—
		8088	—
		80286	5+n
		80386	3
		80486	2
RCL reg,imm RCR reg,imm	RCL CL,2 RCL SI,3 RCR AH,5 RCR BX,13	8086	—
		8088	—
		80286	5+n
		80386	9
		80486	8
ROL mem,imm ROR mem,imm	ROL DATA,4 ROL BYTE PTR [DI],2 ROR NUMB,2 ROR WORD PTR [ECX],3	8086	—
		8088	—
		80286	8+n
		80386	7
		80486	4

RCL mem,imm RCR mem,imm	RCL DATA,6 RCL BYTE PTR [DI],7 RCR NUMB,6 RCR WORD PTR [ECX],5	8086	—
		8088	—
		80286	8+n
		80386	10
		80486	9

REP Repetir prefijo

11110010 1010010w

O D I T S Z A P C

Formato	Ejemplos	Relojes	
REP MOVS	REP MOVSB REP MOVSW REP MOVSD REP MOVS DATA1,DATA2	8086	9+17n
		8088	9+25n
		80286	5+4n
		80386	8+4n
		80486	12+3n

11110010 1010101w

Formato	Ejemplos	Relojes	
REP STOS	REP STOSB REP STOSW REP STOSD REP STOS DATA3	8086	9+10n
		8088	9+14n
		80286	4+3n
		80386	5+5n
		80486	7+4n

11110010 0110110w

Formato	Ejemplos	Relojes	
REP INS	REP INSB REP INSW REP INSD REP INS DATA4	8086	—
		8088	—
		80286	5+4n
		80386	13+6n
		80486	16+8n

11110010 0110111w

Formato	Ejemplos		Relojes
REP OUTS	REP OUTSB REP OUTSW REP OUTSD REP OUTS DATA5	8086	—
		8088	—
		80286	5+4n
		80386	12+5n
		80486	17+5n

REPE/REPNE Repetir condicional

11110011 1010011w — O D I T S Z A P C (Z: *)

Formato	Ejemplos		Relojes
REPE CMPS	REPE CMPSB REPE CMPSW REPE CMPSD REPE CMPS DATA6,DATA7	8086	9+22n
		8088	9+30n
		80286	5+9n
		80386	5+9n
		80486	7+7n

11110011 1010111w

Formato	Ejemplos		Relojes
REPE SCAS	REPE SCASB REPE SCASW REPE SCASD REPE SCAS DATA8	8086	9+15n
		8088	9+19n
		80286	5+8n
		80386	5+8n
		80486	7+5n

11110010 1010011w

Formato	Ejemplos		Relojes
REPNE CMPS	REPNE CMPSB REPNE CMPSW REPNE CMPSD REPNE CMPS DATA9,DATA10	8086	9+22n
		8088	9+30n
		80286	5+9n
		80386	5+9n
		80486	7+7n

11110010 1010111w

Formato	Ejemplos		Relojes
REPNE SCAS	REPNE SCASB REPNE SCASW REPNE SCASD REPNE SCAS DATA11	8086	9+15n
		8088	9+19n
		80286	5+8n
		80386	5+8n
		80486	7+5n

RET Regresar del procedimiento

11000011 — O D I T S Z A P C

Ejemplo		Relojes
RET (cerca)	8086	16
	8088	20
	80286	11
	80386	10
	80486	5

11000010 información

Formato	Ejemplos		Relojes
RET imm (cerca)	RET 4 RET 100H	8086	20
		8088	24
		80286	11
		80386	10
		80486	5

11001011

Ejemplos		Relojes
RET (lejos)	8086	26
	8088	34
	80286	15
	80386	18
	80486	13

11001010 información

Formato	Ejemplos		Relojes
RET imm (lejos)	RET 4 RET 100H	8086	25
		8088	33
		80286	11
		80386	10
		80486	5

SAHF Archivar AH en las banderas

10011110

O	D	I	T	S	Z	A	P	C
				*	*	*	*	*

Ejemplo		Relojes
SAHF	8086	4
	8088	4
	80286	2
	80386	3
	80486	2

SAL/SAR/SHL/SHR Cambiar de posición

1101000w ooTTTmmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	?	*	*

TTT = 100 = SHL/SAL
TTT = 101 = SHR
TTT = 111 = SAR

Formato	Ejemplos		Relojes
SAL reg,1 SHL reg,1 SHR reg,1 SAR reg 1	SAL CL,1 SHL DX,1 SHR CH,1 SAR SI,1	8086	2
		8088	2
		80286	2
		80386	3
		80486	3
SAL mem,1 SHL mem,1 SHR mem,1 SAR mem,1	SAL DATA,1 SHL BYTE PTR [DI],1 SHR NUMB,1 SAR WORD PTR [ECX],1	8086	15+ea
		8088	23+ea
		80286	7
		80386	7
		80486	4

1101001w ooTTTmmm disp

Formato	Ejemplos	Relojes	
SAL reg,CL SHL reg,CL SHR reg,CL SAR reg,CL	SAL CH,CL SHL DX,CL SHR CH,CL SAR SI,CL	8086	8+4n
		8088	8+4n
		80286	5+n
		80386	3
		80486	3
SAL mem,CL SHL mem,CL SHR mem,CL SAR mem,CL	SAL DATA,CL SHL BYTE PTR [DI],CL SHR NUMB,CL SAR WORD PTR [ECX],CL	8086	20+a+4n
		8088	28+a+4n
		80286	8+n
		80386	7
		80486	4

1100000w ooTTTmmm disp información

Formato	Ejemplos	Relojes	
SAL reg,imm SHL reg,imm SHR reg,imm SAR reg,imm	SAL CL,4 SHL DX,5 SHR CH,12 SAR SI,9	8086	—
		8088	—
		80286	5+n
		80386	3
		80486	2
SAL mem,imm SHL mem,imm SHR mem,imm SAR mem,imm	SAL DATA,6 SHL BYTE PTR [DI],7 SHR NUMB,6 SAR WORD PTR [ECX],5	8086	—
		8088	—
		80286	8+n
		80386	7
		80486	4

SBB Restar con pedir prestado

000110dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
SBB reg,reg	SBB CL,DL SBB AX,DX SBB CH,CL SBB EAX,EBX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1

SBB mem,reg	SBB DATA,CL SBB BYTES,CX SBB NUMBS,ECX SBB [EAX],CX	8086	16+ea
		8088	24+ea
		80286	7
		80386	6
		80486	3
SBB reg,mem	SBB CL,DATA SBB CX,BYTES SBB ECX,NUMBS SBB CX,[EDX]	8086	9+ea
		8088	13+ea
		80286	7
		80386	7
		80486	2

100000sw oo011mmm disp información

Formato	Ejemplos		Relojes
SBB reg,imm	SBB CL,4 SBB DX,5 SBB CH,12 SBB SI,9	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
SBB mem,imm	SBB DATA,6 SBB BYTE PTR [DI],7 SBB NUMB,6 SBB WORD PTR [ECX],5	8086	17+ea
		8088	25+ea
		80286	7
		80386	7
		80486	3

0001110w información

Formato	Ejemplos		Relojes
SBB acc,imm	SBB AL,4 SBB AX,5 SBB AH,12 SBB AX,9	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

SCAS Examinar cadena

1010111w

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
SCASB SCASW SCASD	SCASB SCASW SCASD SCAS DATA REP SCASB	8086	15
		8088	19
		80286	7
		80386	7
		80486	6

SET Establecer con condición

00001111 1001cccc oo000mmm

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos	Relojes	
SETcd reg8	SETA BL SETB CH SETG DL SETE BH SETZ AL	8086	—
		8088	—
		80286	—
		80386	4
		80486	3
SETcd mem8	SETE DATA SETLE BYTES	8086	—
		8088	—
		80286	—
		80386	5
		80486	3

Condición Códigos	Mnemonic	Bandera	Descripción
0000	SETO	O = 1	Establecer si existe desbordamiento
0001	SETNO	O = 0	Establecer si no existe desbordamiento
0010	SETB/SETNAE	C = 1	Establecer si está abajo
0011	SETAE/SETNB	C = 0	Establecer si está arriba o igual
0100	SETE/SETZ	Z = 1	Establecer si es igual/cero
0101	SETNE/SETNZ	Z = 0	Establecer si no es igual/no es cero
0110	SETBE/SETNA	C = 1 + Z = 1	Establecer si está abajo o igual
0111	SETA/SETNBE	C = 0 • Z = 0	Establecer si está arriba
1000	SETS	S = 1	Establecer si existe signo
1001	SETNS	S = 0	Establecer si no existe signo
1010	SETP/SETPE	P = 1	Establecer si la paridad es pares
1011	SETNP/SETPO	P = 0	Establecer si la paridad es nones
1100	SETL/SETNGE	S • O	Establecer si es menor que
1101	SETGE/SETNL	S = O	Establecer si es mayor o igual
1110	SETLE/SETNG	Z = 1 + S • O	Establecer si es menor que o igual
1111	SETG/SETNLE	Z = 0 + S = O	Establecer si es mayor

SGDT/SIDT/SLDT Almacenar la tabla del descriptor

00001111 00000001 oo000mmm disp — O D I T S Z A P C

Formato	Ejemplos		Relojes
SGDT mem	SGDT MEMORY SGDT GLOBAL	8086	—
		8088	—
		80286	11
		80386	9
		80486	10

00001111 00000001 oo001mmm disp

Formato	Ejemplos		Relojes
SIDT mem	SIDT DATAS SIDT INTERRUPT	8086	—
		8088	—
		80286	12
		80386	9
		80486	10

00001111 00000000 oo000mmm disp

Formato	Ejemplos		Relojes
SLDT reg	SLDT CX SLDT DX	8086	—
		8088	—
		80286	2
		80386	2
		80486	2
SLDT mem	SLDT NUMBS SLDT LOCALS	8086	—
		8088	—
		80286	3
		80386	2
		80486	3

SHLD/SHRD Cambio de posición con doble precisión

00001111 10100100 oorrrmmm disp información

O	D	I	T	S	Z	A	P	C
?				*	*	?	*	*

Formato	Ejemplos	Relojes	
SHLD reg,reg,imm	SHLD AX,CX,10 SHLD DX,BX,8 SHLD CX,DX,2	8086	—
		8088	—
		80286	—
		80386	3
		80486	2
SHLD mem,reg,imm	SHLD DATA,CX,8	8086	—
		8088	—
		80286	—
		80386	7
		80486	3

00001111 10101100 oorrrmmm disp información

Formato	Ejemplos	Relojes	
SHRD reg,reg,imm	SHRD CX,DX,2	8086	—
		8088	—
		80286	—
		80386	3
		80486	2
SHRD mem,reg,imm	SHRD DATA,CX,3	8086	—
		8088	—
		80286	—
		80386	7
		80486	3

00001111 10100101 oorrrmmm disp

Formato	Ejemplos	Relojes	
SHLD reg,reg,CL	SHLD DX,BX,CL	8086	—
		8088	—
		80286	—
		80386	3
		80486	3

Formato	Ejemplos	Relojes	
SHLD mem,reg,CL	SHLD DATA,AX,CL	8086	—
		8088	—
		80286	—
		80386	7
		80486	3

00001111 10100101 oorrrmmm disp

Formato	Ejemplos	Relojes	
SHRD reg,reg,CL	SHRD DX,BX,CL	8086	—
		8088	—
		80286	—
		80386	3
		80486	3
SHRD mem,reg,CL	SHRD DATA,AX,CL	8086	—
		8088	—
		80286	—
		80386	7
		80486	3

SMSW Almacenar palabra del estado de la máquina

00001111 00000001 oo100mmm disp — O D I T S Z A P C

(sólo se debe usar por el 80286)

Formato	Ejemplos	Relojes	
SMSW reg	SMSW AX SMSW DX SMSW CX	8086	—
		8088	—
		80286	2
		80386	10
		80486	2
SMSW mem	SMSW DATA	8086	—
		8088	—
		80286	3
		80386	3
		80486	3

STC Establecer la bandera de cargar

11111001		O D I T S Z A P C (C = 1)
Ejemplo		Relojes
STC	8086	2
	8088	2
	80286	2
	80386	2
	80486	2

STD Establecer bandera de dirección

11111101		O D I T S Z A P C (D = 1)
Ejemplo		Relojes
STD	8086	2
	8088	2
	80286	2
	80386	2
	80486	2

STI Establecer interruptor para bandera

11111011		O D I T S Z A P C (I = 1)
Ejemplo		Relojes
STI	8086	2
	8088	2
	80286	2
	80386	3
	80486	5

STOS Almacenar información de la cadena

1010101w			O D I T S Z A P C
Formato	Ejemplos		Relojes
STOSB STOSW STOSD	STOSB STOSW STOSD STOS DATA REP STOSB	8086	11
		8088	15
		80286	3
		80386	4
		80486	5

STR Almacenar registro para tareas

00001111 00000000 oo001mmm disp

O D I T S Z A P C

Formato	Ejemplos	Relojes	
STR reg	STR DX STR CX STR AX	8086	—
		8088	—
		80286	2
		80386	2
		80486	2
STR mem	STR DATA	8086	—
		8088	—
		80286	3
		80386	2
		80486	3

SUB Restar

001010dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos	Relojes	
SUB reg,reg	SUB CL,DL SUB AX,DX SUB CH,CL SUB EAX,EBX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
SUB mem,reg	SUB DATA,CL SUB BYTES,CX SUB NUMBS,ECX SUB [EAX],CX	8086	16+ea
		8088	24+ea
		80286	7
		80386	6
		80486	3
SUB reg,mem	SUB CL,DATA SUB CX,BYTES SUB ECX,NUMBS SUB CX,[EDX]	8086	9+ea
		8088	13+ea
		80286	7
		80386	7
		80486	2

100000sw oo101mmm disp información

Formato	Ejemplos		Relojes
SUB reg,imm	SUB CL,4 SUB DX,5 SUB CH,12 SUB SI,9	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
SUB mem,imm	SUB DATA,6 SUB BYTE PTR [DI],7 SUB NUMB,6 SUB WORD PTR [ECX],5	8086	17+ea
		8088	25+ea
		80286	7
		80386	7
		80486	3

0010110w información

Formato	Ejemplos		Relojes
SUB acc,imm	SUB AL,4 SUB AX,5 SUB AH,12 SUB AX,9	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

TEST Operandos de prueba (comparación lógica)

1000011w oorrrmmm disp

O D I T S Z A P C
0 * * ? * 0

Formato	Ejemplos		Relojes
TEST reg,reg	TEST CL,DL TEST CX,DX TEST CL,CH TEST ECX,EBX	8086	5
		8088	5
		80286	2
		80386	2
		80486	1
TEST reg,mem mem,reg	TEST DATA,CL TEST CL,DATA	8086	9+ea
		8088	13+ea
		80286	6
		80386	5
		80486	2

1111011w oo000mmm disp información

Formato	Ejemplos		Relojes
TEST reg,imm	TEST CL,4 TEST DX,5 TEST CH,12H TEST SI,256	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
TEST mem,imm	TEST DATA,6	8086	11+ea
		8088	11+ea
		80286	6
		80386	5
		80486	2

1010100w información

Formato	Ejemplos		Relojes
TEST acc,imm	TEST AL,4 TEST AX,5 TEST AH,12 TEST AX,9 TEST EAX,2	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

VERR/VERW Verificar leer o escribir

00001111 00000000 oo100mmm disp — O D I T S Z A P C (Z: *)

Formato	Ejemplos		Relojes
VERR reg	VERR BX VERR CX VERR DX	8086	—
		8088	—
		80286	14
		80386	10
		80486	11
VERR mem	VERR DATA	8086	—
		8088	—
		80286	16
		80386	11
		80486	11

00001111 00000000 oo101mmm disp

Formato	Ejemplos	Relojes	
VERW reg	VERW AX VERW CX VERW DX	8086	—
		8088	—
		80286	14
		80386	15
		80486	11
VERW mem	VERW DATA	8086	—
		8088	—
		80286	16
		80386	16
		80486	11

WAIT Esperar para el coprocesador

10011011 — O D I T S Z A P C

Ejemplos	Relojes	
WAIT FWAIT	8086	4
	8088	4
	80286	3
	80386	6
	80486	6

WBINVD Escribir e invalidar el cache para información

00001111 00001001 — O D I T S Z A P C

Ejemplo	Relojes	
WBINVD	8086	—
	8088	—
	80286	—
	80386	—
	80486	5

XADD Intercambiar y sumar

0000111p ?!100000w 11rrrrrr

O	D	I	T	S	Z	A	P	C
*				*	*	*	*	*

Formato	Ejemplos		Relojes
XADD reg,reg	XADD EBX,ECX XADD EDX,EAX XADD EDI,EBP	8086	—
		8088	—
		80286	—
		80386	—
		80486	3

00001111 1100000w oorrrmmm disp

Formato	Ejemplos		Relojes
XADD mem,reg	XADD DATA,EAX XADD [DI],EAX XADD [ECX],EDX	8086	—
		8088	—
		80286	—
		80386	—
		80486	4

XCHG Intercambiar

1000011w 1oorrrmmm

O	D	I	T	S	Z	A	P	C

Formato	Ejemplos		Relojes
XCHG reg,reg	XCHG BL,CL XCHG AX,DX XCHG EDI,EBP	8086	4
		8088	4
		80286	3
		80386	3
		80486	3
XCHG reg,mem mem,reg	XCHG CL,DATA XCHG DATA,CL XCHG DX,[DI] XCHG ECX,[EBP]	8086	17+ea
		8088	25+ea
		80286	5
		80386	5
		80486	5

10010reg

Formato	Ejemplos	Relojes	
XCHG acc,reg XCHG reg,acc	XCHG DATA,AL XCHG AX,FRIED	8086	3
		8088	3
		80286	3
		80386	3
		80486	3

XLAT Traducir

11010111

O D I T S Z A P C

Ejemplo	Relojes	
XLAT	8086	11
	8088	11
	80286	5
	80386	3
	80486	4

XOR OR-exclusivo

001100dw oorrrmmm disp

O	D	I	T	S	Z	A	P	C
0				*	*	?	*	0

Formato	Ejemplos	Relojes	
XOR reg,reg	XOR BL,CL XOR CX,DX XOR CH,CL XOR EAX,EBX	8086	3
		8088	3
		80286	2
		80386	2
		80486	1
XOR mem,reg	XOR DATA,CL XOR BYTES,CX XOR NUMBS,ECX XOR [EAX],CX	8086	16+ea
		8088	24+ea
		80286	7
		80386	6
		80486	3

XOR reg,mem	XOR CL,DATA XOR CX,BYTES XOR ECX,NUMBS XOR CX,[EDX]	8086	9+ea
		8088	13+ea
		80286	7
		80386	7
		80486	2

100000sw oo110mmm disp información

Formato	Ejemplos		Relojes
XOR reg,imm	XOR BL,33 XOR CX,234H XOR CH,'A' XOR EAX,123445	8086	4
		8088	4
		80286	3
		80386	2
		80486	1
XOR mem,imm	XOR DATA,34 XOR BYTES,1234 XOR NUMBS,123 XOR [EAX],11	8086	17+ea
		8088	25+ea
		80286	7
		80386	7
		80486	3

0011010w información

Formato	Ejemplos		Relojes
XOR acc,imm	XOR AL,33 XOR AX,234H XOR AL,'A' XOR EAX,123445	8086	4
		8088	4
		80286	3
		80386	2
		80486	1

APENDICE C

Cambios en el bit de bandera

Este apéndice muestra sólo las instrucciones que realmente cambian los bits de la bandera. Cualquier instrucción que no esté mencionada no afectará ninguno de los bits de la bandera.

Instrucción	0	D	I	T	S	Z	A	P	C
AAA	?				?	?	*	?	*
AAD	?				*	*	?	*	?
AAM	?				*	*	?	*	?
AAS	?				?	?	*	?	*
ADC	*				*	*	*	*	*
ADD	*				*	*	*	*	*
AND	0				*	*	?	*	0
ARPL						*			
BSF						*			
BSR						*			
BT									*
BTC									*
BTR									*
BTS									*
CLC									0
CLD		0							
CLI			0						
CMC									*

Instrucción	*Banderas*								
	O	D	I	T	S	Z	A	P	C
CMP	*				*	*	*	*	*
CMPS	*				*	*	*	*	*
CMPXCHG	*				*	*	*	*	*
DAA	?				*	*	*	*	*
DAS	?				*	*	*	*	*
DEC	*				*	*	*	*	*
DIV	?				?	?	?	?	?
IDIV	?				?	?	?	?	?
IMUL	*				?	?	?	?	*
INC	*				*	*	*	*	*
IRET	*	*	*	*	*	*	*	*	*
LAR					*				*
LSL					*				*
MUL	*				?	?	?	?	*
NEG	*				*	*	*	*	*
OR	0				*	*	?	*	0
POPF/POPFD	*	*	*	*	*	*	*	*	*
RCL/RCR	*								*
REPE/REPNE						*			
ROL/ROR	*								*
SAHF					*	*	*	*	*
SAL/SAR	*				*	*	?	*	*
SHL/SHR	*				*	*	?	*	*
SBB	*				*	*	*	*	*
SCAS	*				*	*	*	*	*
SHLD/SHRD	?				*	*	?	*	*
STC									1
STD		1							
STI			1						
SUB	*				*	*	*	*	*
TES	0				*	*	?	*	*
VERR/VERW					*				
XADD	*				*	*	*	*	*
XOR	0				*	*	?	*	0

APENDICE D

Normas (estándares) para canales

Este apéndice ilustra los estándares para un canal que se encuentra en la mayor parte de los sistemas copiados e IBM-PC. Por lo regular a estos canales se les llama ISA (arquitectura estándar de IBM). Aunque existen otros estándares de canal, éstos representan a los dos que son los que más a menudo se encuentran en los sistemas de computadoras.

La figura D-1 ilustra el estándar ISA de 8-bits que se encuentra en la computadora XT. La figura D-2 ilustra el estándar ISA de 16-bits que se encuentra en la computadora AT.

Observe que el conector superior en el estándar AT es idéntico al estándar XT. Esto permite que tarjetas de 8 bits más viejas sean conectadas a cualquier ranura AT de 16-bits en un sistema de computadora tipo AT.

FIGURA D–1 El conector de borde para el tipo-XT de 8-bits.

Parte posterior de la computadora

GND	B1	A1	$\overline{\text{IO CHCK}}$
RESET DRV	B2	A2	SD7
+5VDC	B3	A3	SD6
IRQ9	B4	A4	SD5
-5VDC	B5	A5	SD4
DRQ2	B6	A6	SD3
-12VDC	B7	A7	SD2
OWS	B8	A8	SD1
+12VDC	B9	A9	SD0
GND	B10	A10	$\overline{\text{IO CHRDY}}$
$\overline{\text{SMEMW}}$	B11	A11	AEN
$\overline{\text{SMEMR}}$	B12	A12	SA19
IOW	B13	A13	SA18
$\overline{\text{IOR}}$	B14	A14	SA17
$\overline{\text{DACK3}}$	B15	A15	SA16
DRQ3	B16	A16	SA15
$\overline{\text{DACK1}}$	B17	A17	SA14
DRQ1	B18	A18	SA13
$\overline{\text{REFRESH}}$	B19	A19	SA12
CLK	B20	A20	SA11
IRQ7	B21	A21	SA10
IRQ6	B22	A22	SA9
IRQ5	B23	A23	SA8
IRQ4	B24	A24	SA7
IRQ3	B25	A25	SA6
$\overline{\text{DACK2}}$	B26	A26	SA5
T/C	B27	A27	SA4
BALE	B28	A28	SA3
+5VDC	B29	A29	SA2
OSC	B30	A30	SA1
GND	B31	A31	SA0

Conector del componente lateral del borde

FIGURA D–2 El conector de canal para el tipo-AT de 16-bits.

Parte posterior de la computadora

GND	B1	A1	IO CHCK
RESET DRV	B2	A2	SD7
+5VDC	B3	A3	SD6
IRQ9	B4	A4	SD5
-5VDC	B5	A5	SD4
DRQ2	B6	A6	SD3
-12VDC	B7	A7	SD2
OWS	B8	A8	SD1
+12VDC	B9	A9	SD0
GND	B10	A10	IO CHRDY
SMEMW	B11	A11	AEN
SMEMR	B12	A12	SA19
IOW	B13	A13	SA18
IOR	B14	A14	SA17
DACK3	B15	A15	SA16
DRQ3	B16	A16	SA15
DACK1	B17	A17	SA14
DRQ1	B18	A18	SA13
REFRESH	B19	A19	SA12
CLK	B20	A20	SA11
IRQ7	B21	A21	SA10
IRQ6	B22	A22	SA9
IRQ5	B23	A23	SA8
IRQ4	B24	A24	SA7
IRQ3	B25	A25	SA6
DACK2	B26	A26	SA5
T/C	B27	A27	SA4
BALE	B28	A28	SA3
+5VDC	B29	A29	SA2
OSC	B30	A30	SA1
GND	B31	A31	SA0

MEMCS16	D1	C1	SBHE
IOCS16	D2	C2	LA23
IRQ10	D3	C3	LA22
IRQ11	D4	C4	LA21
IRQ12	D5	C5	LA20
IRQ15	D6	C6	LA19
IRQ14	D7	C7	LA18
DACK0	D8	C8	LA17
DRQ0	D9	C9	MEMR
DACK5	D10	C10	MEMW
DRQ5	D11	C11	SD08
DACK6	D12	C12	SD09
DRQ6	D13	C13	SD10
DACK7	D14	C14	SD11
DRQ7	D15	C15	SD12
+5VDC	D16	C16	SD13
MASTER	D17	C17	SD14
GND	D18	C18	SD15

Conector del componente lateral del borde

APENDICE E

Respuestas a las preguntas y problemas con un número par

CAPITULO 1

2. Los video juegos, hornos de microondas y sistemas de control sencillos.
4. Las velocidades de ejecución del microprocesador de 4 bits fueron de 20 KIPS, con el microprocesador más actualizado de 32 bits alcanzando 60 MIPs.
6. 4G bytes.
8. Una canalización es un término que se utiliza para describir el flujo de la información por el microprocesador. El microprocesador ejecuta el software más eficientemente porque la canalización alimentará a diversas unidades dentro del microprocesador con las instrucciones en varias fases de ejecución.
10. 1M
12. 32 bits
14. 16 bits
16. No existe ninguna diferencia entre los mapas de la memoria.
18. 512K bytes
20. 100000H
22. 8. AL, AH, BL, CL, CH, DL y DH.
24. 4. EAX, EBX, ECX y EDX.
26. Se utiliza con ciertas instrucciones para almacenar la información para la ejecución por la instrucción.
28. El lado más a la derecha del registro para segmento es incrementado con 0000_2 u OH.
30. Sí, 16 bytes.
32. 12440H
34. Segmento para información. Segmento adicional.
36. Z = bandera cero indica cuando el resultado es cero
    S = bandera de signo sostiene el signo (+ o -)
    C = mantiene el restante después de sumar o pedir prestado después de restar
    A = mantiene medio restante o es medio prestado
    P = mantiene la paridad (pares o nones)

38. La bandera D selecciona autoincremento o autoreducción para las instrucciones de la cadena.
40. 8 bits, 16 bits y 32 bits.
42. Palabra: 10000H = 34H y 10001H = 12H.
    Doble palabra: 10000H = 34H, 10001H = 12H, 10002 = 00H y 10003 = 00H.
44. Cualquier registro para segmento.
46. (a) 1111111110010111, (b) 0000000100101110, (c) 1111111111110100, (d) 0000000010000110 y (e) 1111000101000101.
48. (a) 0 10000010 01000000000000000000000
    (b) 1 10000010 01100000000000000000000
    (c) 0 10000101 10010000100000000000000
    (d) 1 10000101 00000100010000000000000
    (e) 0 10000111 00101100000110000000000

CAPITULO 2

2. AL, AH, BL, BH, CL, CH, DL Y DH.
4. EAX, EBX, ECX, EDX, ESP, EBP, ESI Y EDI.
6. No puede utilizar un registro de tamaños mixtos.
8. (a) MOV EDX,EBX
   (b) MOV CL,BL
   (c) MOV BX,SI
   (d) MOV AX,DS
   (e) MOV AH,AL
10. #
12. Un modo de indicar para direccionar información de la memoria.
14. (a) 03234H, (b) 02300H y (c) 02400H
16. MOV BYTE PTR [DI],3
18. MOV DWORD PTR [BX],23456H
20. (a) 21100H, (b) 10100H y (c) 21000H
22. (a) 11750H, (b) 11950H y (c) 11700H
24. BP
26. Directo, relativo e indirecto.
28. El cambio de intersegmentos permite que el programa se mueva a cualquier lugar de la memoria, mientras que el cambio intrasegmento le permite moverse a cualquier localidad dentro del segmento del código actual.
30. Un cambio lejano le permite al programa moverse a cualquier localidad de la memoria.
32. (a) corto, (b) cercano, (c) corto y (d) lejano.
34. JMP NEAR PTR [BX] asumiendo que BX contiene una TABLA de direcciones desplazadas.
36. La instrucción PUSH [DI] obtiene una palabra de la localidad de la memoria del segmento de información direccionada por DI y la empuja a la pila.
38. La instrucción PUSHAD empuja a EAX, EBX, ECX, EDX, ESP, EBP, EDI y ESI a la pila.

CAPITULO 3

2. D = la dirección del flujo de la información y W indica una palabra o doble palabra.
4. DL
6. [BX+DI]
8. MOV AX,BX
10. 8B77
12. No puede cambiar el registro para el segmento del código.
14. CS
16. EAX, ECX, EDX, EBX, ESP, EBP, ESI y EDI.
18. La instrucción PUSH BX copia el contenido de BX a la pila. 020FFH = BH y 020FEH = BL.
20. 2
22. La instrucción DI,NUMB carga a DI con los 16-bits de información archivada en la localidad de la memoria NUMB. La instrucción LEA DI,NUMB almacena la dirección de NUMB en DI.
24. MOV con OFFSET es más eficiente.
26. Ambas instrucciones son idénticas, excepto que LDS cargará al registro DS y LSS cargará al registro SS.
28. Selecciona el modo de autoincremento o autorreducción para las instrucciones de la cadena.
30. DI direcciona información en el segmento adicional y SI direcciona información en el segmento de datos.
32. La instrucción STOSW copia AX a la localidad de la memoria del segmento adicional direccionado por DI y aumentará o reducirá DI por 2.
34. El prefijo REP (repetir) repite una instrucción de cadena el número de veces cargada al registro CX.
36. En el registro DX.
38. En el programa traducido de un microprocesador 8085.
40. (véase el ejemplo E-1)

EJEMPLO E–1

```
TABLE   DB    30H,31H,32H,33H
        DB    33H,35H,36H,37H
        DB    38H,39H
        MOV   BX,OFFSET TABLE
        XLAT
```

42. La instrucción OUT DX,AX envía el contenido de AX a un periférico de E/S direccionado por DX.
44. MOV ES:[BX],AH.
46. Un comando al ensamblador que puede o no generar información.
48. LIST1 DB 30 DUP (?)
50. El directivo ,386 selecciona el conjunto de instrucciones 80386 para el ensamblador.
52. Modelos de memoria.
54. Termina un programa y regresa el control a DOS.
56. Permite que el registro seleccionado sea automáticamente empujado arriba de la pila o eliminada de la pila dentro de un procedimiento.
58. (véase el ejemplo E-2)

EJEMPLO E-2

```
COPY    PROC FAR

        MOV  AX,CS:DATA1
        MOV  BX,AX
        MOV  CX,AX
        MOV  DX,AX
        MOV  SI,AX
        RET

COPY    ENDP
```

CAPITULO 4

2. No puede utilizar tamaños mixtos de registro/información.
4. Sum = 2100H. Z = 0, C = 0, AC = 1, P = 1, S = 0 y O = 0.
6. (véase el ejemplo E-3)

EJEMPLO E-3

```
ADD  AX,BX
ADD  AX,CX
ADD  AX,DX
ADD  AX,SP
MOV  DI,AX
```

8. ADC DX,BX
10. El ensamblador no tiene manera de identificar la información de la memoria como un byte, palabra, o palabra doble.
12. Diferencia = 81H, Z = 0, C = 0, AC = 0, P = 1, S = 1, O = 0.
14. DEC EBX
16. SUB y CMP realizan restas y las banderas cambian para cada instrucción, pero CMP descarta el resultado.
18. DX y AX.
20. EDX y EAX.
22. (véase el ejemplo E-4)

EJEMPLO E-4

```
MOV  AL,DL
MUL  DL
MUL  DL
MOV  DX,AX
```

24. AX
26. División por cero y desbordamiento.
28. AH

30. DAA y DAS.
32. AAM divide el contenido de AX por 10 y deja al resultado en AH y el restante de AL.
34. (a) AND BX,DX, (b) AND DH,OEAH, (c) AND DI,BP, (d) AND EAX,1122H, (e) AND [BP],CX y AND DX,[SI-8].
36. (a) OR AH,BL, (b) OR ECX,88H, (c) OR SI,DX, (d) OR BP,1122H, (e) OR [BX],CX, (f) OR AL,[BP+40] y OR WHEN,AH.
38. (a) XOR AH,BH, (b) XOR CL,99H, (c) XOR DX,DI, (d) XOR ESP,1A23H, (e) XOR [EBX],DX, (f) XOR DI,[BP+60], y (g) XOR DI,WELL.
40. Tanto AND y TEST realizan la operación AND y cambian las banderas, pero el resultado será descartado por la instrucción TEST.
42. La instrucción NOT realiza el complemento de los 1s, mientras que la instrucción NEG realizará el complemento de los 2s.
44. La instrucción SCASB comparará a AL con la localidad de la memoria del segmento adicional direccionado por el registro DI.
46. La bandera D = 0 para seleccionar autoincremento y la bandera D = 1 para seleccionar autoreducción para los apuntadores SI y/o DI de la instrucción de la cadena.
48. La instrucción REPNE SCASB termina si CX = 0 o si AL = el contenido de la localidad de memoria de segmento adicional direccionado por DI.
50. (véase el ejemplo E-5)

EJEMPLO E-5

```
MOV   DI,OFFSET LIST
MOV   CX,300H
MOV   AL,66H
REPNE SCASB
```

CAPITULO 5

2. El salto cercano.
4. El salto lejano.
6. Es un nivel cercano.
8. El registro IP en los registros IP y CS.
10. La instrucción JMP DI salta a la ubicación de la memoria direccionada por DI y el JMP [DI] va a la localidad de la memoria del segmento de información direccionado por DI y saca la dirección del cambio de esta localidad.
12. Z (JNZ/JNE o JZ/JE), S (JS O JNS), C (JC o JNC), P (JP o JNP), O (JO o JNO).
14. Siempre que el bit de la bandera de desbordamiento esté establecido.
16. JZ, JNZ, JA, JAE, JB o JBE.
18. JCXZ salta si CX = 0000H.
20. CX
22. La instrucción LOOPE se reduce a CX y luego prueba al bit de la bandera cero y se cambia si la condición es cero. Observe que la reducción no afecta a los bits de la bandera.
24. (véase el ejemplo E-6)

EJEMPLO E-6

```
LOOK    PROC NEAR

        MOV  SI,OFFSET BLOCK
        MOV  CX,100H
        MOV  UP,0
        MOV  DOWN,0
REPS:
        CMP  BYTE PTR [SI],42H
        JA   REPS1
        JB   REPS2
        LOOP REPS
        RET
REPS1:
        INC  UP
        LOOP REPS
        RET
REPS2:
        INC  DOWN
        LOOP REPS
        RET

LOOK    ENDP
```

26. El CALL cercano llama a un procedimiento dentro del segmento para código actual, mientras que el CALL lejano llama a un procedimiento de cualquier localidad de la memoria.
28. RET
30. Un procedimiento es identificado como cercano o lejano utilizando NEAR y FAR con el comando PROC.
32. (véase el ejemplo E-7)

EJEMPLO E-7

```
CUBE    PROC NEAR
        XOR  DX,DX
        MOV  AX,CX
        MUL  CX
        MUL  CX
        RET

CUBE    ENDP
```

34. (véase el ejemplo E-8)

EJEMPLO E-8

```
SUMS    PROC NEAR

        XOR  EDI,EDI
        ADD  EAX,EBX
        JNC  SUMS1
        INC  EDI
```

```
SUMS1:
        ADD   EAX,ECX
        JNC   SUMS2
        INC   EDI
SUMS2:
        ADD   EAX,EDX
        JNC   SUMS3
        INC   EDI
SUMS3:
        RET

SUMS    ENDP
```

36. INT o INTO.
38. Error de división.
40. La instrucción IRET saca las banderas de la pila además de regresar.
42. Cuando el desbordamiento está establecido.
44. STI y CLL
46. Interruptor para vector 9.
48. Cuando el valor del registro no está dentro de la frontera superior e inferior.
50. BP

CAPITULO 6

2. TEST.OBJ, TEST.LST y TEST.CRF.
4. Indica que la etiqueta se puede utilizar por otros módulos de programación.
6. BYTE PTR, WORD PTR, DWORD PTR, NEAR PTR y FAR PTR.
8. MACRO
10. Los parámetros son transferidos al macro ya sea en registro o como información especial que sigue al comando MACRO.
12. El macro LOCAL debe seguir inmediatamente al comando MACRO e identifica los variables locales.
14. (véase el ejemplo E-9)

EJEMPLO E–9

```
ADDM    MACRO  LIST,LENGTH
        LOCAL ADM1,ADM2
        CLD
        MOV   CX,LENGTH
        MOV   SI,OFFSET LIST
        XOR   AX,AX
ADM1:
        ADD   AL,[SI]
        JNC   ADM2
        INC   AH
ADM2:
        INC   SI
        LOOP  ADM1
        ENDM
```

16. (véase el ejemplo E-10)

EJEMPLO E-10

```
RANDOM  PROC NEAR

        INC  CL
        MOV  AH,6
        MOV  DL,0FFH
        INT  21H
        JZ   RANDOM
        RET

RANDOM  ENDP
```

18. (véase el ejemplo E-11)

EJEMPLO E-11

```
READ_E  PROC NEAR

        MOV  AH,6
        MOV  DL,0FFH
        INT  21H
        JE   READ_E
        OR   AL,AL
        JNE  READ_E
        INT  21H
        CALL DISP
        ROR  AL,1
        ROR  AL,1
        ROR  AL,1
        ROR  AL,1
        CALL DISP
        RET

READ_E  ENDP

DISP    PROC NEAR

        PUSH AX
        AND  AL,0FH
        ADD  AL,30H
        CMP  AL,39
        JBE  DISP1
        ADD  AL,7
DISP1:
        MOV  DL,AL
        INT  21H
        POP  AX
        RET

DISP    ENDP
```

20. AAM
22. 30H

24. (véase el ejemplo E-12)

EJEMPLO E-12

```
READ    PROC  NEAR

        MOV   DI,OFFSET DATA
READ1:
        MOV   AH,6
        MOV   DL,0FFH
        INT   21H
        JZ    READ1
        CMP   AL,30H
        JB    READ2
        CMP   AL,39H
        JA    READ2
        SUB   AL,30H
        STOSB
        JMP   READ1
READ2:
        RET

READ    ENDP
```

26. (véase el ejemplo E-13)

EJEMPLO E-13

```
CONVERT PROC NEAR

        CMP  AL,'a'
        JB   CONVERT1
        CMP  AL,'z'
        JA   CONVERT1
        SUB  AL,20H
CONVERT1:
        RET

CONVERT ENDP
```

28. (véase el ejemplo E-14)

EJEMPLO E-14

```
CMP  AL,6
JE   ONE
CMP  AL,7
JE   TWO
CMP  AL,8
JE   THREE
```

30. El sector de arranque contiene un programa que carga a DOS en la memoria del sistema. El FAT asignará los sectores del disco a varios archivos del programa. El directorio raíz contiene 128 nombres de archivo o subdirectorio.

32. El arrancador cargará a DOS en la memoria y se encuentra en el sector de arranque.
34. Si es sólo para lectura, una etiqueta de volumen, etc.
36. 4,294,967,295 bytes.
38. (véase el ejemplo E-15)

EJEMPLO E-15

```
RENAME  PROC NEAR

        PUSH ES
        MOV  AX,DS
        MOV  ES,AX
        MOV  DI,OFFSET TEST_LST
        MOV  EI,OFFSET TEST_LIS
        MOV  AH,56H
        INT  21H
        POP  ES
        RET

RENAME  ENDP
```

40. (véase el ejemplo E-16)

EJEMPLO E-16

```
DATA1   DB   13,10,'2 = $'
DATA2   DB   13,10,'$'
POWERS  PROC NEAR
        MOV  CX,8
POWERS1:
        MOV  AH,9
        MOV  DX,OFFSET DATA1
        INT  21H
        PUSH CX
        MOV  AX,CX
        SUB  AX,8
        MOV  CX,AX
        MOV  AL,1
        ROR  AL,CL
        CALL DISP
        POP  CX
        MOV  AH,9
        MOV  DX,OFFSET DATA2
        INT  21H
        MOV  AX,CX
        SUB  8
        CALL DISP
        LOOP POWERS1
POWERS  ENDP
DISP    PROC NEAR
        XOR  AH,AH
        AAM
        CMP  AH,0
        JZ   DISP1
        PUSH AX
        ADD  AH,30H
        MOV  DL,AH
```

```
        MOV  AH,6
        INT  21H
        POP  AX
DISP1:
        ADD  AL,30H
        MOV  DL,AL
        MOV  AH,6
        INT  21H
        RET
DISP    ENDP
```

42. (véase el ejemplo E-17)

EJEMPLO E-17 (página 1 de 4)

```
PROG    SEGMENT
        ASSUME CS:PROG

MAIN    PROC FAR

        CALL GET_ADR
        CALL NEW
        CALL DISP
        MOV  AH,4CH
        INT  21H

MAIN    ENDP

MES1    DB   13,10,'Escriba la dirección de arranque: $'

GET_ADR PROC NEAR

        PUSH DS
        MOV  AX,CS
        MOV  DS,AX
        MOV  DX,OFFSET MES1
        MOV  AH,9
        INT  21H
        POP  DS
        CALL READ
        RET

GET_ADR ENDP

READ    PROC NEAR

        XOR  CX,CX
READ1:
        CALL GET
        CMP  AL,13                  ;tecla de enter
        JE   READ3
        CMP  AL,'0'
        JB   READ1
        CMP  AL,'9'
        JBE  READ2
        SUB  AL,7
        CMP  AL,'A-7'
        JB   READ1
```

EJEMPLO E–17 (página 2 de 4)

```
        CMP  AL,'F-7'
        JA   READ1
READ2:
        SUB  AL,'0'
        PUSH AX
        INC  CX
        JMP  READ1
READ3:
        XOR  DX,DX
        XOR  BX,BX
        OR   CX,CX
        JE   READ5
READ4:
        POP  AX
        SHL  BX,1
        RCL  DX,1
        SHL  BX,1
        RCL  DX,1
        SHL  BX,1
        RCL  DX,1
        SHL  BX,1
        RCL  DX,1
        ADD  BL,AL
        LOOP READ4
        SHR  DX,1
        RCR  BX,1
        SHR  DX,1
        RCR  BX,1
        SHR  DX,1
        RCR  BX,1
        SHR  DX,1
        RCR  BX,1
READ5:
        MOV  DS,BX
        XOR  SI,0
        RET

READ    ENDP

NEW     PROC NEAR

        MOV  AL,13
        CALL DIP
        MOV  AL,10
        CALL DIP
        RET

NEW     ENDP

DIP     PROC NEAR

        MOV  AH,6
        MOV  DL,AL
        INT  21H
        RET

DIP     ENDP
```

EJEMPLO E-17 (página 3 de 4)

```
GET     PROC NEAR

        MOV  AH,6
        MOV  DL,0FFH
        INT  21H
        JE   GET
        RET

GET     ENDP

DISP    PROC NEAR

        MOV  CX,256
DISP1:
        MOV  AX,SI
        AND  AX,0FH
        JNZ  DISP2
        CALL ADDR
DISP2:
        LODSB
        CALL DIPS
        MOV  AL,20H
        INT  21H
        LOOP DISP1
        RET

DISP    ENDP

DIPS    PROC NEAR

        PUSH AX
        SHR  AX,4
        ADD  AL,30H
        CMP  AL,'9'
        JBE  DIPS1
        ADD  AL,7
DIPS1:
        CALL DIP
        POP  AX
        ADD  AL,30H
        CMP  AL,'9'
        JBE  DIPS2
        ADD  AL,7
DIPS2:
        CALL DIP
        RET

DIPS    ENDP

ADDR    PROC NEAR

        CALL NEW
        MOV  AX,DS
        PUSH AX
        MOV  AL,AH
        CALL DIPS
        POP  AX
        CALL DIPS
        MOV  AL,':'
        CALL DIPS
```

EJEMPLO E–17 (página 4 de 4)

```
        MOV  AX,SI
        MOV  AL,AH
        CALL DIPS
        MOV  AX,SI
        CALL DIPS
        MOV  AL,20H
        CALL DIPS
        RET

ADDR    ENDP

PROG    ENDS

        END  MAIN
```

CAPITULO 7

2. El 8086/8088 es compatible con TTL siempre que la corriente de carga esté limitada a no más de 2.0 mA.
4. A7-A0
6. Que el microprocesador está a punto de leer información por las conexiones del canal de información.
8. La señal aplicada a la entrada de información de CLK debe ser una onda cuadrada compatible con TTL que tenga una frecuencia entre 500 KHz y 5.0 MHz para la versión estándar.
10. El WR indica que el microprocesador está a punto de escribir información por medio de su canal de información al E/S o sistema de memoria.
12. La señal DT/R es una lógica uno para redirigir a los búferes para canal para transmitir información del microprocesador a la memoria y sistema E/S.
14. IO/M, DT/R, y SSO.
16. Los bits del estado de la cola de espera indican la condición de la cola de espera interna para el coprocesador numérico externo.
18. 3
20. 2.333 MHz
22. AD15-AD0, A19/S6-A16/S3 y BHE/S7.
24. El sujetador 74LS373.
26. A causa de que el 8086/8088 sólo utiliza 2.0 mA de corriente, necesitamos a menudo un búfer al microprocesador para incrementar la corriente del lector de disco.
28. 4
30. De leer o escribir.
32. (a) utilizado para proporcionar a la memoria o E/S con la dirección
    (b) permite tiempo para que la memoria o E/S tengan acceso a información
    (c) asigna las señales de control para leer o escribir
    (d) lee o escribe la información
34. El DEN es 580 ns de ancho.
36. Selecciona una etapa de sincronización.
38. La operación del modo mínimo es parecida a los anteriores microprocesadores de 8-bits, en donde la máxima operación del modo será proporcionada para la interface del microprocesador a su coprocesador.

CAPITULO 8

2. (a) 256, (b) 2K, (c) 4K y (d) 8K.
4. Estos pins habilitan o seleccionan al equipo de la memoria.
6. El WE es utilizado para habilitar una operación de escribir si el pin CS también está activado.
8. El 8088 permite 460 ns de tiempo de acceso para la memoria, cuando incluye el tiempo que se requiere para decodificar la dirección de la memoria, menos de 450 ns sobran para acceso de la memoria. Esto previene que los 450 ns EPROM trabajen correctamente.
10. El pin S selecciona al periférico de la memoria para que pueda leer o escribir información, el pin G ocasiona leer cuando el pin S está activo, y el pin W ocasiona escribir cuando el pin S está activo.
12. Memoria acceso aleatoria dinámico.
14. El CAS envía la dirección de la columna a los pins de la dirección múltiplex del DRAM y RAS envía la dirección desde la hilera a los pins.
16. Los decodificadores de la dirección de la memoria permiten que más de un periférico para la memoria funcione en el sistema de la memoria.
18. (véase la figura E-1)
20. (véase la figura E-2)



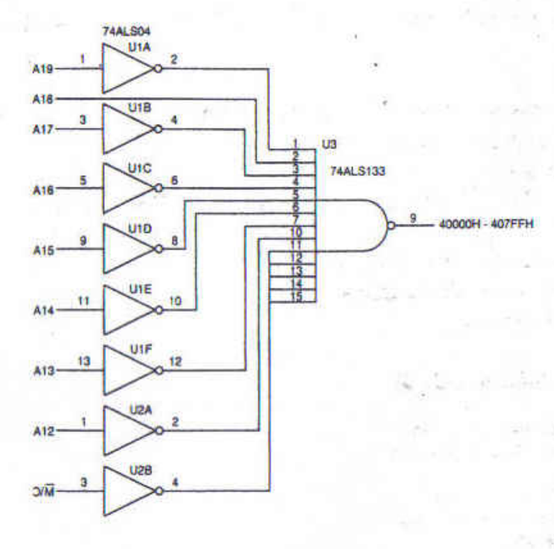
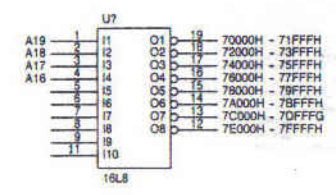


FIGURA E-2



22. El decodificador 74LS139 es un decodificador dual de 2 a 4 líneas.

24.

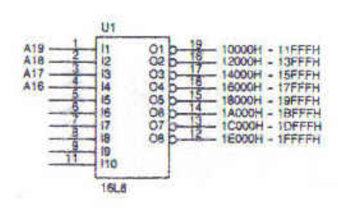
Entradas										Salidas							
OE	A8	A7	A6	A5	A4	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7
0	0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	0	0	0	0	1	1	0	1	1	1	1	1	1
0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	1
0	0	0	1	0	0	0	0	1	1	1	1	1	0	1	1	1	1
0	0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	1	1
0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	1	1
0	0	0	1	0	0	0	1	1	0	1	1	1	1	1	1	0	1
0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	0
todas las demás combinaciones de direcciones										1	1	1	1	1	1	1	1

26. $\overline{\text{MRDC}}$ y $\overline{\text{MWTC}}$
28. (véase la figura E-3)
30. (véase la figura E-4)
32. 5
34. 1
36. El $\overline{\text{BHE}}$ indica que el microprocesador utilizará D15-D8 para escribir o leer y A0 indica que usará A7-A0.
38. alto
40. El microprocesador puede no leer parte del canal, pero esto no causará ningún daño si la información es presentada a esa parte del canal con el $\overline{\text{RD}}$.
42. (véase la figura E-5)
44. Un solo ciclo de $\overline{\text{RAS}}$ será usado para refrescar al DRAM sin leer o escribir información.
46. 15.625 µs.
48. Esto selecciona un banco de memoria.
50. TWST inserta un estado de espera.
52. (véase el ejemplo E-18)

EJEMPLO E-18 (página 1 de 3)

```
TITULO      Address Decoder
PATRON      Test 1U4 (PAL U4)
REVISION    A
AUTOR       Barry B. Brey
COMPAÑIA    Symbiotic Systems
FECHA       6/27/93
CHIP        Decoder1U4 PAL16L8
```

FIGURA E-3



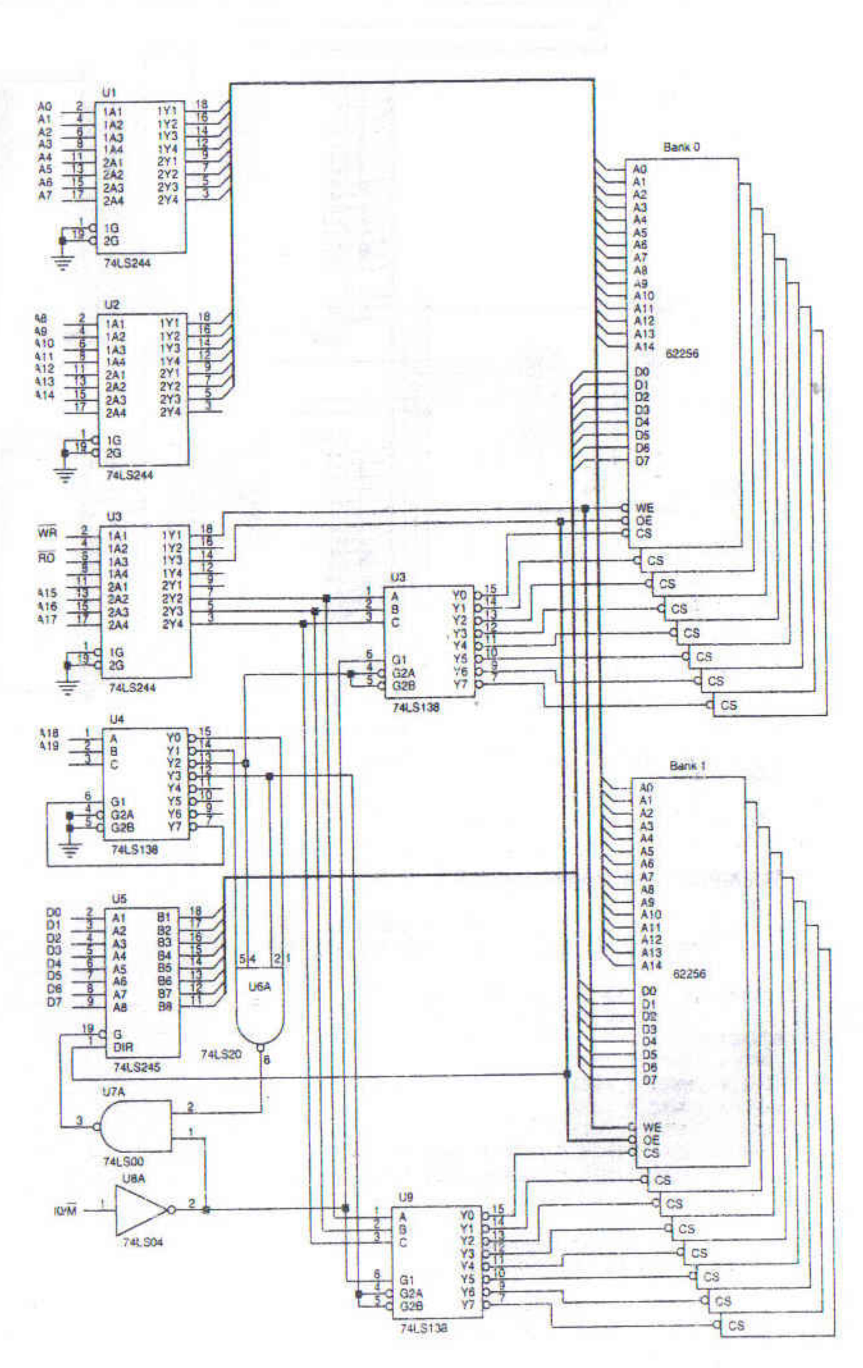

U1
74LS244
U2
74LS244
U3
74LS244
U4
74LS138
U5
74LS245
U6A
74LS20
U7A
74LS00
U8A
74LS04
U9
74LS138
Bank 0
Bank 1
62256
IO/M
WR
RD

FIGURA E–4

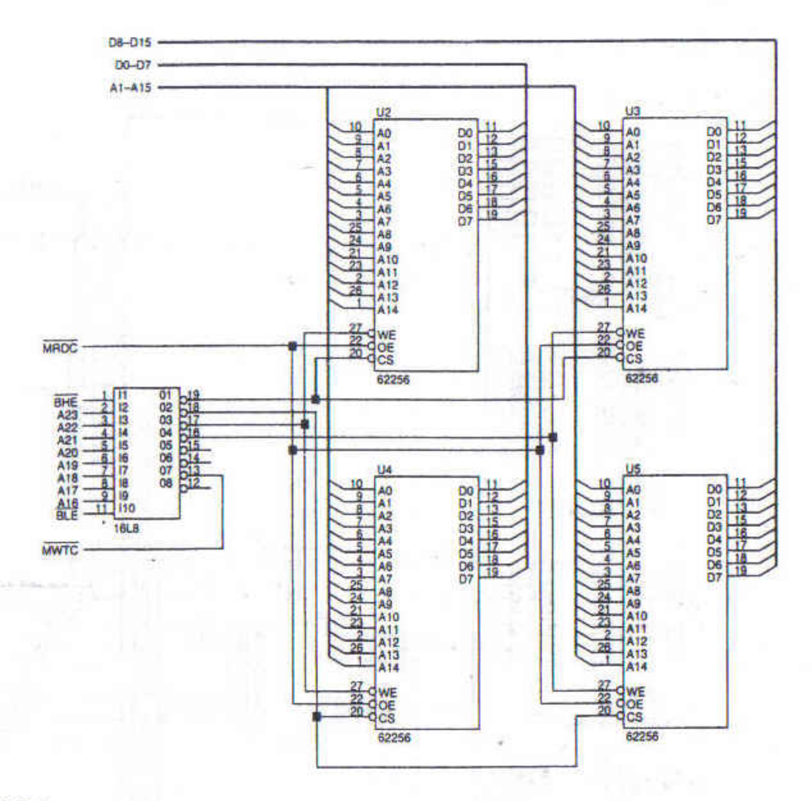


FIGURA E–5

EJEMPLO E–18 (página 2 de 3)

```
;pins 1     2   3   4   5   6   7   8   9   10
      MWTC BE0 BE1 BE2 BE3 A22 A23 A24 A25 GND

;pins  11  12 13 14  15  16  17  18  19  20
       A26 NC U2 MB1 WR0 WR1 WR2 WR3 MB0 VCC
ECUACIONES
/WR0 = /MWTC * /BE0
/WR1 = /MWTC * /BE1
/WR2 = /MWTC * /BE2
/WR3 = /MWTC * /BE3
/MB0 = /A26 * /A25 * /A24 * /A23 * /A22 * /U2
/MB1 = /A26 * /A25 * /A24 * /A23 * A22 * /U2


TITULO       Address Decoder
PATRON       Test 1U5 (PAL U5)
REVISION     A
AUTOR        Barry B. Brey
COMPAÑIA     Symbiotic Systems
```

EJEMPLO E–18 (página 3 de 3)

```
FECHA          6/27/93
CHIP           Decoder1U5 PAL16L8

;pins 1   2   3   4   5   6   7  8  9  10
      A27 A28 A29 A30 A31 CAS NC NC NC GND

;pins  11 12 13 14 15 16 17 18 19 20
       NC NC NC NC NC NC NC NC U2 VCC

EQUATIONS

/U2 = A31 * /A30 * A29 * /A28 * /A27 * /CAS
```

CAPITULO 9

2. El byte después del código de operación mantiene el número del puerto.
4. DX.
6. La instrucción OUTSB copia el contenido de la localidad de la memoria del segmento de información direccionado por SI al puerto E/S direccionado por DX. Después de la transferencia, SI será incrementado por uno o reducido por uno.
8. El E/S aislado utiliza los puertos E/S ubicados en un mapa separado llamado un *mapa E/S* en la dirección 0000H-FFFFH para tener acceso a E/S. La distribución de memoria de E/S utiliza una porción del sistema de la memoria para tener acceso a los periféricos E/S.
10. El interface para salida de información básico es un decodificador del puerto E/S conectado a la entrada de información del reloj de un pasador que capta la información del canal de datos, siempre que la instrucción de salida de información es ejecutada.
12. bajo.
14. (véase la figura E-6)
16. (véase la figura E-7)
18. (véase la figura E-8)
20. D7-D0.
22. 24.
24. A1 y A0.
26. (véase la figura E-9)
28. Pasador de E/S, E/S de strobe y E/S bidireccional.
30. La armadura rota cada vez que hay un flujo de corriente por medio de un par de bobinas.

FIGURA E–6

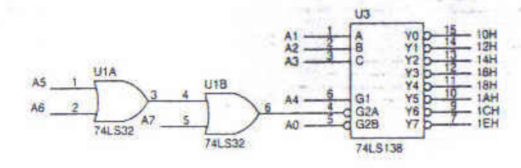


FIGURA E–7

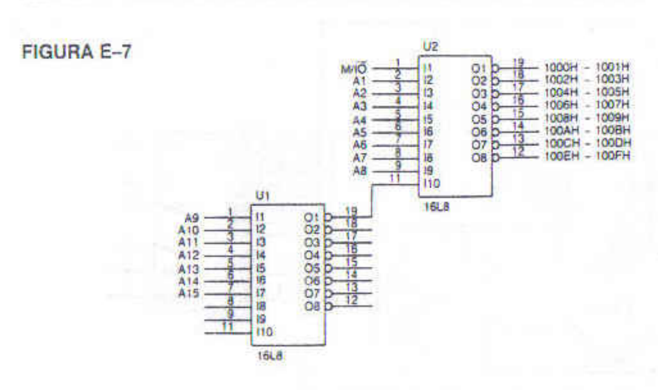


FIGURA E–8

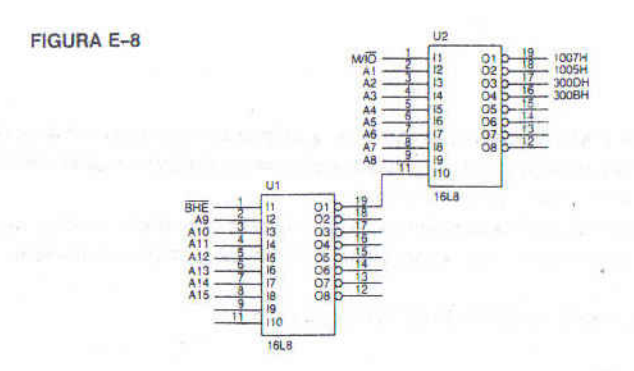
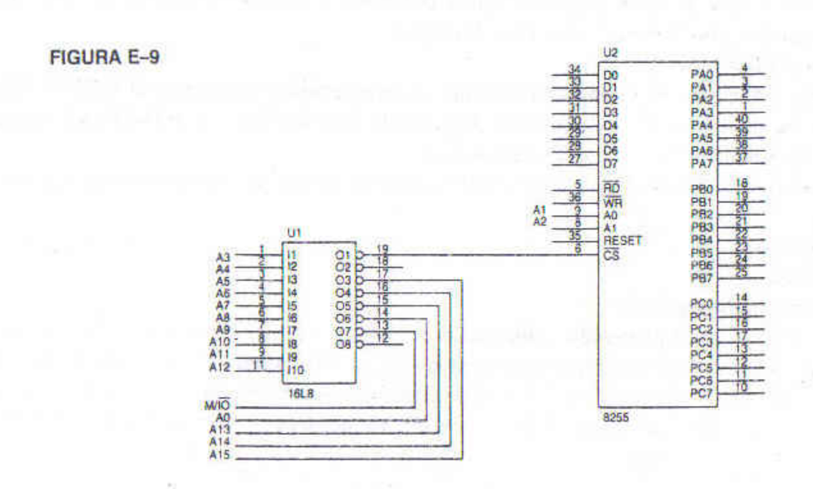


FIGURA E–9



32. (véase el ejemplo E-19)

EJEMPLO E–19

```
MOV  AL,0FH
OUT  COMMAND,AL
```

34. La señal $\overline{ACK}$ borra la bandera completa del búfer para salida de información para indicar que los datos han sido removidos del puerto E/S.
36. (véase el ejemplo E-20)

EJEMPLO E–20

```
IN   AL,PORTC
TEST AL,10H
JNZ  PC4_IS_ONE
```

38. PC0-PC2.
40. 2
42. Un error de desbordamiento ocurre cuando el software falla en quitar información del teclado FIFO cuando se llena.
44. (véase la figura E-10)
46. 10 MHz
48. (véase la figura E-11)
50. Byte de orden bajo.
52. (véase ejemplo E-21)

EJEMPLO E–21

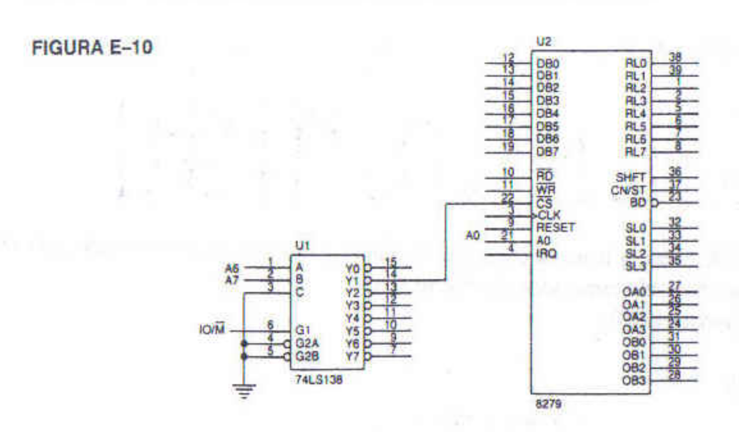
```
;Este software supone un reloj de 1 MHz
;
MOV  AL,74H
OUT  COMMAND,AL              ;seleccionar modo 2
MOV  AL,65H                  ;el conteo es 101
OUT  TIMER1,AL
XOR  AL,AL
OUT  TIMER1,AL
```

54. Información que será transmitida sin una señal de reloj.
56. El número de bits transmitidos por segundo incluyendo los bits de arranque y de paro.
58. 36,000 Hz.
60. Enviando tres 00Hs al registro para comando seguido por el comando de rearrancar 40H.
62. +.01 V
64. (véase el ejemplo E-22)

EJEMPLO E–22

```
MOV  DX,400H                 ;dirección del puerto 400H
XOR  AL,AL                   ;iniciar información en 00H
```

FIGURA E–10



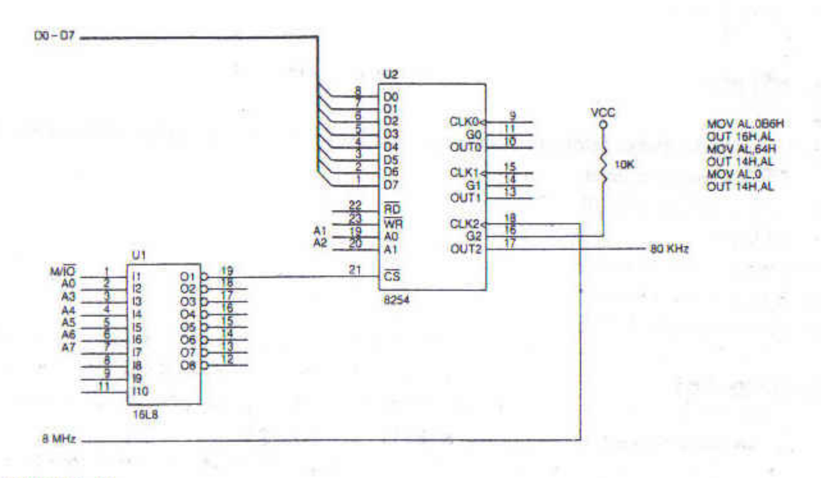
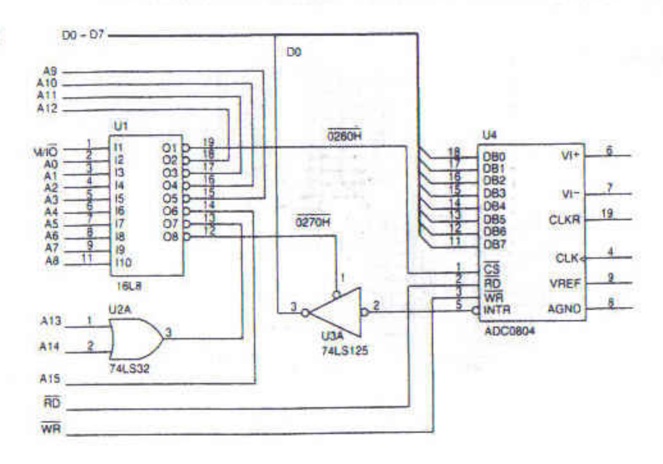


FIGURA E–11

```
REPS:
        OUT  DX,AL                      ;enviar datos
        CALL DELAY                      ;esperar 17.7 µs
        INC  AL
        JNZ  REPS
        DEC  AL
REPS1:
        DEC  AL
        JZ   REPS
        OUT  DX,AL
        CALL DELAY
        JMP  REPS1
```

66. Este pin señala el fin de la conversión.
68. (véase la figura E-12)

FIGURA E–12



CAPITULO 10

2. Un interruptor es un CALL iniciado con hardware a un procedimiento que le da servicio al interruptor. Este CALL interrumpe al programa actualmente ejecutándose hasta que el procedimiento del servicio del interruptor esté completo.
4. Los periféricos E/S no requieren atención hasta que activen la entrada de información del requerimiento del interruptor al microprocesador. Esto utiliza la menor cantidad de tiempo de ejecución.
6. INT nn (incluyendo a INT 3), INTO, IRET, STI y CLI.
8. 00000H-003FFH
10. Vectores 0-31 (0H-1FH).
12. La instrucción BOUND compara un registro con el contenido de dos palabras de la memoria, una contiene el límite inferior y la otra el límite superior. Si el número en el registro está fuera de estos "límites," ocurre un interruptor.
14. 00110H-00113H.
16. Este interruptor es utilizado para emular al coprocesador numérico en el 80286-80486.
18. La bandera del interruptor controla el pin INTR. Si la bandera = 0, el INTR está escondido y si la bandera = 1, el pin INTR está activo.
20. La bandera del interruptor será borrada y establecida con las instrucciones CLI y STI.
22. 2
24. Positivo
26. Vector
28. (véase la figura E-13)
30. Cuando el microprocesador utiliza estas conexiones del canal de información para determinar el número del interruptor para vector, verá un FFH porque el canal está ubicado alto.
32. La cadena de margarita no indica cuál periférico ocasionó el interruptor; ésta es responsabilidad del software, el cual debe encuestar a los periféricos para determinar cuál periférico causó el interruptor.
34. 9

FIGURA E-13

36. Los pins en cascada son utilizados cuando más de un 8259 está presente en el sistema.
38. La ICW es una palabra para un comando de iniciación utilizada para programar al 8259.
40. ICW1, ICW2 e ICW4.
42. ICW1
44. Un comando que termina al interruptor actualmente activo como es determinado por el 8259A.
46. El IRR indica cuáles niveles del interruptor están escondidos.

CAPITULO 11

2. El microprocesador suspende la ejecución de la instrucción actual y flota su dirección, información y canales para control.
4. E/S a la memoria.
6. DACK
8. El microprocesador es ideal y el controlador DMA gobierna al sistema.
10. 4
12. Registro para comandos.
14. (véase el ejemplo E-23)

EJEMPLO E-13

```
MOV  AL,2H
OUT  LATCH,AL           ;establecer A-19-A16 a 2
OUT  FL,AL              ;borrar F/L FF
XOR  AL,AL              ;fuente de 21000H
OUT  CHO_ADD
MOV  AL,10H
OUT  CHO_ADD
XOR  AL,AL              ;destinación de 20000H
OUT  CH1_ADD,AL
OUT  CH1_ADD,AL
MOV  AL,0FFH            ;cuenta del programa
OUT  CH1_CNT,AL
XOR  AL,AL
```

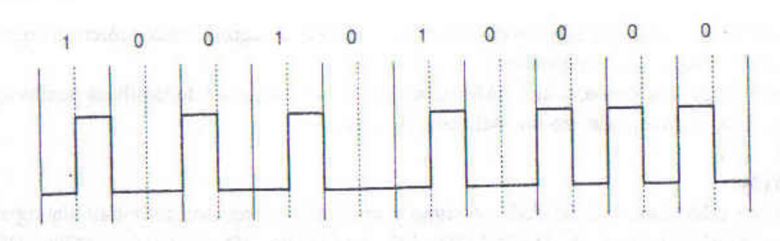


FIGURA E-14

```
OUT   CH1_CNT,AL
MOV   AL,88H              ;modo del programa
OUT   MODE,AL
MOV   AL,85H
OUT   MODE,AL
MOV   AL,1
OUT   CMD,AL              ;establecer movimiento de bloque
MOV   AL,0EH
OUT   MASKS,AL            ;definir canal 0
MOV   AL,4
OUT   REQ,AL              ;requerir DMA
```

16. Mini
18. Pistas
20. Cilindro
22. (véase la figura E-14)
24. El conjunto de la cabeza en un disco fijo está diseñada aerodinámicamente para deslizarse en un cojín de aire muy delgado sobre la superficie del disco. Se le llama una cabeza voladora.
26. El resorte de voz es mucho más exacto que el motor de pasos porque su posición puede ser finamente ajustada mientras que el sistema del disco opera.
28. Un CDROM es muy semejante al audio CD en que almacena información en la forma de música o códigos digitales para la computadora.
30. Rojo, verde y azul.
32. La unidad más pequeña mostrada en un sistema de pantalla de vídeo.
34. El monitor TTL RGB muestra 16 colores porque controla los tres colores primarios de luz a dos intensidades.
36. 2,097,152 colores.
38. 32,400/60 = 540.

CAPITULO 12

2. Los números enteros de palabras fluctúan en valor de −32,768 a +32,767, el número entero corto de $-2 \times 10^{+9}$ a $+2 \times 10^{-9}$, y el número entero largo de $-9 \times 10^{+18}$ a $+9 \times 10^{-18}$.
4. Precisión sencilla (32 bits), precisión doble (64-bits) y precisión extendida (80 bits).
6. (a) − 15, (b) + .5625, (c) + 306, (d) + 1, (e) 10, (f) = + 0.0.
8. Al pin $\overline{\text{BUSY}}$ en el 80286.

10. El 80287 es ideal para ejecutar una instrucción de un coprocesador mientras que el 80286 ejecuta instrucciones normales.
12. Después de una instrucción FXAM indican si la parte superior de la pila es positiva, negativa, inválida, normalizada, no normalizada, 0 o vacía.
14. FLDCW
16. ST(0).
18. Affine selecciona la infinidad con signo y projective selecciona infinidad sin signo.
20. (a) FROG DQ 23.44, (b) DATA3 DD –123, (c) DATA1 DD –23.8 y (d) DATA2 DQ?.
22. Carga a un número entero de la localidad DATA de la memoria arriba de la pila.
24. La instrucción FADD agrega la parte superior de la pila al próximo número arriba de la pila ST(1) y elimina ambos números de la pila. El resultado será colocado arriba de la pila.
26. Almacena al ST como un número BCD en los 80 bits de la memoria comenzando con DATA y después saca la parte de arriba de la pila.
28. Ambos verifican la parte superior de la pila, el FTST compara al ST con 0, mientras que la instrucción FXAM reporta lo que está arriba de la pila. Los bits del estado del registro contienen (C0-C3) esta información.
30. FLDPI
32. FSTENV
34. (véase el ejemplo E-24)

EJEMPLO E–24

```
FLD  L
FLD  W
FMUL
FSTP A
```

36. (véase el ejemplo E-25)

EJEMPLO E–25

```
        MOV  SI,OFFSET TABLE          ;tabla
        MOV  NUMB,2                   ;número entero de 16-bits
REPS:
        FILD NUMB                     ;obtener número
        FSQRT
        FSTP DWORD PTR [SI]           ;guardar la raiz cuadrada
        ADD  SI,4
        INC  NUMB
        CMP  NUMB,11
        JNE  REPS
```

38. (véase el ejemplo E-26)

EJEMPLO E–26

```
FLD1
FDIV  R2                  ;1/R2
FLD1
```

```
FDIV  R3            ;1/R3
FLD1
FDIV  R4            ;1/R4
FADD                ;1/R3 + 1/R4
FADD                ;1/R2 + 1/R3 + 1/R4
FLD1
FDIV
FADD  R1
FSTP  RT
```

CAPITULO 13

2. Generador del reloj, tres medidores de tiempo, controlador para interruptor programable, controlador de DMA programable y una unidad de selección de chips de memoria/periféricos.
4. 10 MHz
6. 2.0 mA
8. La señal ALE aparece antes por una mitad de periodo del reloj.
10. 417 ns
12. (véase ejemplo E-27)

EJEMPLO E–27

```
MOV  AX,3100H       ;nuevo valor de relocalidad
MOV  DX,0FFFEH
OUT  DX,AX
```

14. 9
16. El registro para control selecciona prioridad, nivel de activación, definición y tipo de operación para el periférico.
18. Si el estado de la encuesta del interruptor es leído, el interruptor será reconocido; si el registro para la encuesta del interruptor es leído, el interruptor no será reconocido.
20. 3
22. Timer 2
24. Controla si un EN puede o no cambiar.
26. Selecciona un solo registro para la cuenta máxima (A) o una operación para un registro dual para la cuenta máxima (A más B).
28. (véase el ejemplo E-28)

EJEMPLO E–28

```
MOV  AX,123
MOV  DX,0FF5AH
OUT  DX,AX          ;programar A
MOV  AX,23
ADD  DX,2
OUT  DX,AX          ;programar B
MOV  DX,0FF58H
MOV  AX,8007H
OUT  DX,AX          ;comenzar Timer 1
```

30. 2
32. El canal es arrancado por el software (registro para control) o por hardware (medidor de tiempo No. 2 o la entrada de información DRQ).
34. 7
36. Base
38. 0 y 3
40. Selecciona la operación de $\overline{PCS5}$/A0 y $\overline{PCS6}$/A1.
42. IG
44. Verifica si una operación de lectura se puede realizar en un segmento de la memoria.

CAPITULO 14

2. 64T
4. (véase la figura E-15)
6. El sistema de la memoria es de 32 bits de ancho y contiene cuatro bancos que son de 8 bits de ancho. Las señales para selección del banco (BEO-BE3) seleccionan los cuatro bancos para una operación de 32-bits, dos bancos para una operación de 16-bits y un solo banco para una operación de 8-bits.
8. El canalizador alarga el tiempo de acceso porque el microprocesador envía antes a la dirección una pulsación de reloj.
10. 0000H-FFFFH

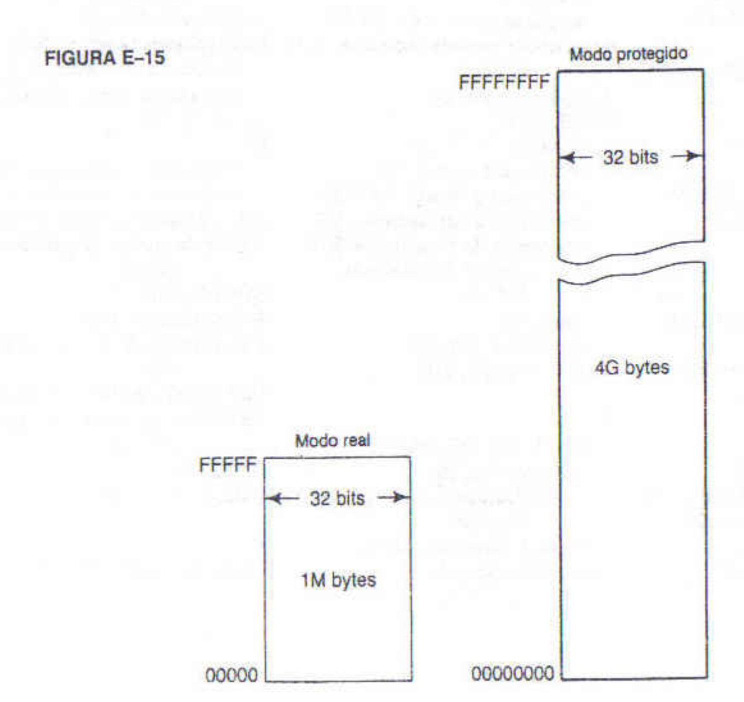


FIGURA E-15

12. El espacio E/S del 80386 es idéntico, pero la memoria contiene 4,096 bytes en vez de 1M byte en el 8086/8088.
14. Esto estructura al 80386 para que tenga un canal de información de 16 bits.
16. EAX, EBX, ECX, EDX, ESP, EBP, ESI, EDI, EIP y EFLAGS.
18. CR0 controla el ambiente del modo protegido y selecciona la paginación. CR1 no se utiliza, CR2 mantiene la dirección predeterminada de la página cuando la paginación está activa y CR3 sostiene la dirección base para el directorio de páginas.
20. Tipo 1
22. La instrucción para examinar a la inversa los bits revisará un registro buscando un uno lógico de la izquierda a la derecha; si alguno es encontrado, su bit para la posición de bit será reportado.
24. MOV FS:[DI],EAX
26. Si
28. Actúa como un interruptor para la emulación del coprocesador.
30. Ocurre una falla doble del interruptor cuando un interruptor de tipo 10, 11, 12 o 13 ocurre al mismo tiempo que uno de tipo 14.
32. Un descriptor indica la localidad, tamaño y derechos de acceso de un segmento de la memoria, un interruptor, procedimiento o una tarea.
34. El bit TI en el registro para segmento es una lógica 1.
36. 8,192
38. Un descriptor de segmento describe un código, información o segmento de pila y un descriptor de sistema describe un interruptor, solicitud o movimiento.
40. El TSS será direccionado cargando un valor nuevo al TR (registro para tareas).
42. El 80386 cambia del modo protegido al modo real colocando una lógica 0 en el bit PE de CR0.
44. Con la dirección base en CR3.
46. La unidad de la paginación reasignará la dirección de la memoria D0000000H a la localidad C0000000H mientras que una instrucción se ejecuta.
48. El FLUSH borra el contenido del cache 80486 interno.
50. El registro de la bandera 80486 contiene un bit de la bandera AC para uso con el coprocesador 80487.
52. Par
54. 16 bytes
56. El término indica que un escrito ocurre para el caché y la memoria simultáneamente.
58. Si, el caché es deshabilitado por un bit en CR0.
60. Copia AL a AL o CL a CL.
62. Deshabilita al caché.

INDICE